MACHINE LEARNING

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

Support Vector Machine (Part 2: Soft Margin)

Hard Margin SVM

Nhắc lại:

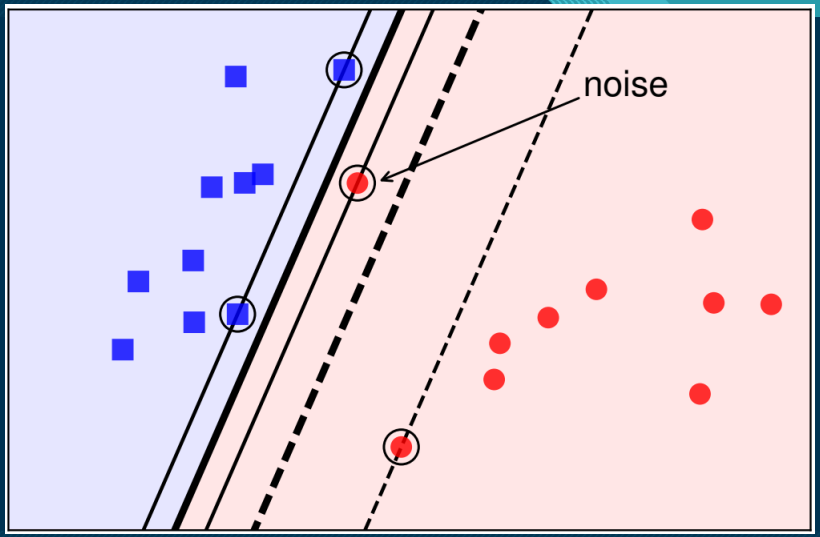
- Support Vector Machine (trong phần 1) cần dữ liệu được phân thành 02 lớp tách được tuyến tính - *linearly separable*.
- Thực tế để tồn tại margin > 0, dữ liệu cho SVM cần tách chặt.
- Với trường hợp dữ liệu gần như tách được tuyến tính - *nearly linearly separable*, SVM không thực hiện được do không tồn tại margin.

Vì những lý do trên, phương pháp SVM nguyên bản còn được gọi là SVM - Lề cứng (**Hard margin**).

- Một cách tự nhiên, chúng ta cũng mong muốn SVM có thể áp dụng với dữ liệu gần như tách được tuyến tính - nearly linearly separable, giống như Logistic Regression.
- Minh họa tiếp theo cho thấy một số khả năng dữ liệu dẫn đến SVM không thực hiện hoặc không hiệu quả.

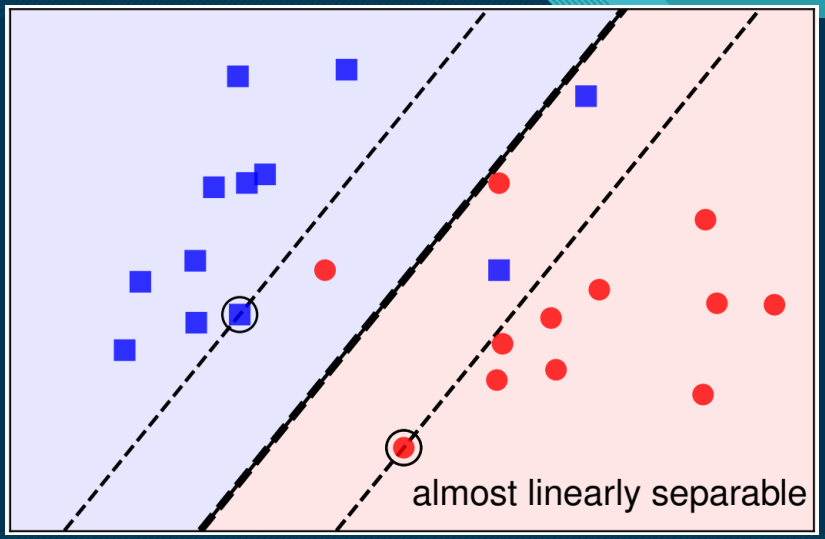
Hard Margin SVM's imperfection

Hình 1a: SVM không hiệu quả do margin rất nhỏ chỉ vì 01 điểm nhiễu



Hard Margin SVM's imperfection

Hình 1b: SVM không thực hiện được với dữ liệu gần như tách được tuyến tính



Hard Margin SVM's imperfection

Phân tích các trường hợp SVM không hiệu quả hoặc thậm chí không làm việc

- **Trường hợp 1**: Dữ liệu vẫn linearly separable như Hình 1a), nhưng có một điểm nhiễu của lớp tròn đỏ ở quá gần với lớp vuông xanh.
  - Nếu sử dụng SVM nguyên bản thì sẽ tạo ra một margin rất nhỏ.
  - Xét tổng thể dữ liệu, đường phân lớp nằm quá gần lớp vuông xanh và xa lớp tròn đỏ.
  - Trong khi đó, nếu ta bỏ qua điểm nhiễu này thì sẽ thu được margin tốt hơn rất nhiều (mô tả bởi các đường nét đứt).
  - Cũng vì vậy SVM nguyên bản được coi là nhạy cảm với nhiễu (*sensitive to noise*).

Hard Margin SVM's imperfection

Phân tích các trường hợp SVM không hiệu quả hoặc thậm chí không làm việc

- **Trường hợp 2**: Dữ liệu chỉ gần như tách được tuyến tính (*nearly linearly separable*) như Hình 1b).
  - SVM nguyên bản dẫn đến bài toán tối ưu không có phương án (tập phương án là rỗng - *infeasible*), do đó vô nghiệm.
  - SVM nguyên bản không thực hiện được trong trường hợp này. Tuy nhiên, nếu bỏ qua một số ít điểm ở gần biên giữa hai classes, ta lại thu được một siêu phẳng tách tốt (đường nét đứt đậm).
  - Lúc đó, các đường margin support nét đứt mảnh tạo được một margin lớn cho bộ phân lớp này.
  - Với mỗi điểm nằm sai phía so với các đường suport (hay *đường margin*, hoặc *đường biên*) tương ứng, ta nói điểm đó rơi vào **vùng không an toàn**.

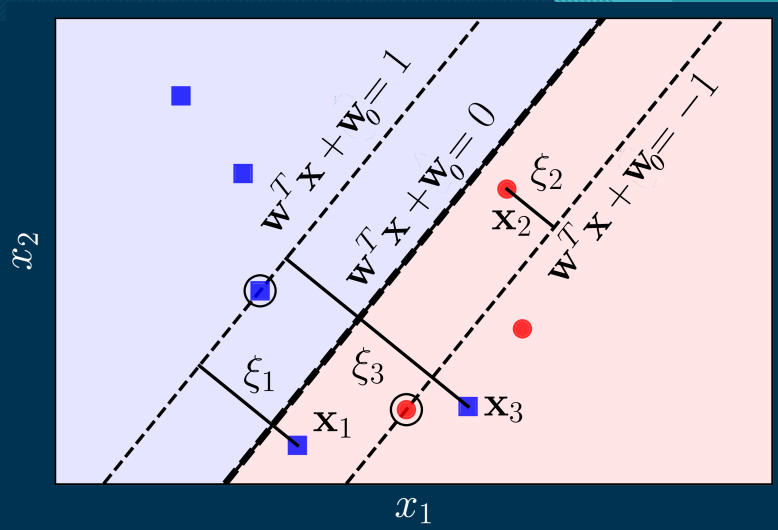
Soft vs Hard Margin

- Trong cả hai hình vẽ, margin tạo bởi đường phân chia và đường nét đứt mảnh còn được gọi là *soft margin* (biên mềm).
- Theo cách gọi này, SVM nguyên bản còn được gọi là *Hard Margin SVM* (SVM biên cứng).
- Từ phân tích trên, để có margin lớn hơn trong SVM, một cách tự nhiên cách tiếp cận sẽ dựa vào việc cân bằng hai yêu cầu
  - Hy sinh (bỏ qua) một số điểm dữ liệu bằng việc chấp nhận một số điểm sẽ rơi vào vùng không an toàn.
  - Hạn chế việc bỏ qua nói trên bằng việc áp đặt một lượng phạt, mỗi điểm bị bỏ qua sẽ làm gia tăng một lượng phạt lên hàm mục tiêu. Do đó mô hình sẽ không cho phép bỏ qua quá nhiều điểm.
- Vậy hàm mục tiêu trong phương pháp SVM cải biên này cần phải là sự kết hợp để cực đại hóa margin nhưng cực tiểu hóa lượng phạt, tức cực tiểu hóa số điểm phải hy sinh.

Soft Margin

- Có hai cách tiếp cận Bài toán tối ưu cho *Soft Margin SVM*.
  1. **Cách thứ nhất**: Giải bài toán tối ưu có ràng buộc thông qua bài toán đối ngẫu (giống như *Hard Margin SVM*).
  2. **Cách thứ hai**: Đưa về một bài toán tối ưu không ràng buộc. Bài toán này có thể giải bằng các phương pháp Gradient Descent.
- Cách thứ nhất cho phép phát triển SVM cho các tập dữ liệu không thực sự tách được tuyến tính (non-linearly separable), ví dụ *Kernel SVM*.
- Cách thứ hai có thể được áp dụng cho các bài toán *large scale*. Ngoài ra, cách này cũng cho phép mở rộng SVM cho bài toán *multi-class classification* (*Multi-class SVM*)

Optimization problem for Soft Margin SVM



Optimization problem for Soft Margin SVM

- Trước hết ta xây dựng Bài toán tối ưu cho *Soft Margin SVM* bằng cách
  1. Tìm siêu phẳng tách và các điểm - vector tựa (support vectors) với việc chấp nhận bỏ qua (hy sinh) - cho phép một số điểm nằm vào vùng *không an toàn*.
  2. Bổ sung vào hàm tổn thất đại lượng khống chế số điểm rơi vào vùng không an toàn thông qua lượng phạt: Nhiều điểm rơi vào vùng không an toàn thì lượng phạt tăng lên.
- Bổ sung biến *slack variable* đo lượng bỏ qua ở vùng không an toàn ξ_n ứng với mỗi điểm dữ liệu x_n:
  - Với siêu phẳng tách hiện tại, nếu x_n nằm trong vùng an toàn, $\xi_n = 0$.
  - Nếu x_n nằm trong vùng không an toàn và giả sử $y_n \in \{\pm 1\}$ là nhãn của nó. Lúc đó lượng phạt là khoảng cách từ x_n đến siêu phẳng lề (siêu phẳng qua các điểm tựa - điểm gần nhất), và dễ dàng xác định được công thức (tự chứng minh):

$$\xi_n = \left|\mathbf{w}^T \mathbf{x}_n + w_0 - y_n\right|$$

Optimization problem for Soft Margin SVM

- Nhắc lại bài toán tối ưu cho Hard Margin SVM:

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

- Với Soft Margin SVM, ta cộng thêm lượng *slack variable* nói trên cho tất cả các điểm trong vùng không an toàn và được hàm tổn thất mới

$$J(\mathbf{w}, w_0) = \frac{1}{2}||\mathbf{w}||_2^2 + C\sum_{n=1}^{N}\xi_n$$

trong đó $C > 0$ là hằng số điều chỉnh mức độ quan trọng giữa độ lớn của margin và số điểm rơi vào vùng không an toàn (lượng hy sinh); còn $\xi = [\xi_1, \xi_2, \ldots, \xi_N]$.

Optimization problem for Soft Margin SVM

- Ràng buộc cho bài toán tối ưu của Soft Margin SVM - với mỗi cặp điểm (x_n, y_n):

$$y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 - \xi_n \Leftrightarrow 1 - \xi_n - y_n(\mathbf{w}^T\mathbf{x}_n + b) \leq 0, \quad \forall n = 1, 2, \ldots, n$$

và ràng buộc phụ $\xi_n \geq 0, \ \forall n = 1, 2, \ldots, N.$

- Bài toán tối ưu cho Soft Margin SVM:

$$(\mathbf{w}, w_0) = \arg\min_{\mathbf{w}, w_0} J(\mathbf{w}, w_0) = \arg\min_{\mathbf{w}, w_0} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{n=1}^{N}\xi_n \tag{2}$$

$$\text{subject to:} \quad 1 - \xi_n - y_n(\mathbf{w}^T\mathbf{x}_n + w_0) \leq 0$$
$$-\xi_n \leq 0, \forall n = 1, 2, \ldots, N.$$

Optimization problem for Soft Margin SVM

Nhận xét

- Nếu C nhỏ, nghiệm bài toán (2) sẽ tập trung vào việc cực đại hóa margin. Ngược lại, nếu C lớn, nghiệm có thể làm margin nhỏ nhưng hạn chế số điểm không an toàn.
- Nếu C rất lớn và dữ liệu thực sự tách được tuyến tính, ta có thể thu được $\sum_{n=1}^{N} \xi_n = 0$, tức không có điểm ở vùng không an toàn.
- Nếu $\xi_n = 0$, điểm x_n thuộc vùng an toàn; nếu $0 < \xi_n < 1$, điểm x_n thuộc vùng không an toàn nhưng được phân loại đúng; $\xi_n > 1$, điểm x_n bị phân loại sai.
- Bài toán tối ưu (2) vẫn là một bài toán lồi, hơn nữa nó có thể biểu diễn dưới dạng một Quadratic Programming (QP - quy hoạch toàn phương).

Lagrange duality of Soft Margin SVM Opt. prob.

- **Kiểm tra tiêu chuẩn Slater**:
- Với mọi $n = 1, 2, \ldots, N$ và với mọi $(\mathbf{w}, w_0)$, luôn tồn tại $\xi_n \geq 0$, $n = 1, 2, \ldots, N$ - đủ lớn để

$$y_n(\mathbf{w}^T \mathbf{x}_n + w_0) + \xi_n > 1, \ \forall n = 1, 2, \ldots, N$$

- vậy bài toán (2) thỏa mãn tiêu chuẩn Slater.

Lagrange duality of Soft Margin SVM Opt. prob.

- **Bài toán Lagrange**:

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

- trong đó $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_N]^T \succeq 0$ và $\mu = [\mu_1, \mu_2, \ldots, \mu_N]^T \succeq 0$ là các biến đối ngẫu Lagrange (vector nhân tử Lagrange) tương ứng của $(\mathbf{w}, w_0)$ và ξ.

Lagrange duality of Soft Margin SVM Opt. prob.

- Hàm số đối ngẫu của bài toán tối ưu (2) là:

$$g(\lambda, \mu) = \min_{\mathbf{w}, w_0, \xi} \mathcal{L}(\mathbf{w}, w_0, \xi, \lambda, \mu).$$

- Với mỗi cặp (λ, μ), chúng ta cần xét $(\mathbf{w}, w_0, \xi)$ thoả mãn điều kiện đạo hàm của Lagrangian bằng 0:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \Leftrightarrow \mathbf{w} = \sum_{n=1}^{N} \lambda_n y_n \mathbf{x}_n \tag{4}$$

$$\frac{\partial \mathcal{L}}{\partial w_0} = 0 \Leftrightarrow \sum_{n=1}^{N} \lambda_n y_n = 0 \tag{5}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_n} = 0 \Leftrightarrow \lambda_n = C - \mu_n, \quad n = 1, 2, \ldots, N. \tag{6}$$

Lagrange duality of Soft Margin SVM Opt. prob.

- Thay các biểu thức (4) - (6) vào Lagrangian ta sẽ thu được hàm đối ngẫu:

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

- chú ý μ có vai trò ẩn trong phiếm hàm này thông qua đẳng thức (6). Vậy nếu viết gộp ràng buộc lên λ: $0 \leq \lambda_n \leq C$ ta có thể viết lại bài toán đối ngẫu

$$\begin{aligned} \lambda &= \arg\max_{\lambda} g(\lambda) \\ \text{subject to: } & \sum_{n=1}^{N} \lambda_n y_n = 0 \\ & 0 \leq \lambda_n \leq C, \ \forall n = 1, 2, \ldots, N. \end{aligned} \tag{7}$$

Lagrange duality of Soft Margin SVM Opt. prob.

- Do μ có vai trò ẩn trong các ràng buộc của (7) nên bài toán này gần giống với bài toán đối ngẫu của Hard Margin SVM. Điểm khác nhau là ở bài toán mới λ_n bị chặn trên

$$0 \leq \lambda_n \leq C, \ \forall n = 1, 2, \ldots, N.$$

- Khi giá trị của C rất lớn, ta có thể coi hai bài toán là như nhau.
- Bài toán này cũng hoàn toàn giải được bằng các công cụ giải QP thông thường, ví dụ CVXOPT như trong bài minh họa cho phương pháp Hard Margin SVM.
- Sau khi tìm được λ của bài toán đối ngẫu, ta vẫn phải quay lại tìm nghiệm $(\mathbf{w}, w_0, \xi)$ của bài toán ban đầu.

KKT system of Soft Margin SVM Opt. prob.

- Tương tự trường hợp Hard margin, có thể kiểm tra dễ dàng tiêu chuẩn Slater cho bài toán trong trường hợp Soft margin.
- Xét hệ điều kiện KKT của bài toán tối ưu Soft Margin SVM

$$1 - \xi_n - y_n(\mathbf{w}^T\mathbf{x}_n + w_0) \leq 0 \tag{8}$$
$$-\xi_n \leq 0 \tag{9}$$
$$\lambda_n \geq 0 \tag{10}$$
$$\mu_n \geq 0 \tag{11}$$
$$\lambda_n(1 - \xi_n - y_n(\mathbf{w}^T\mathbf{x}_n + b)) = 0 \tag{12}$$
$$\mu_n\xi_n = 0 \quad \text{với mọi} \quad n = 1, 2, \ldots, N \tag{13}$$

KKT system of Soft Margin SVM Opt. prob.

- Ta bổ sung thêm các ràng buộc trên λ_n vào $\mathbf{w}$ vào hệ điều kiện KKT

$$\mathbf{w} = \sum_{n=1}^{N} \lambda_n y_n \mathbf{x}_n \tag{4}$$

$$\sum_{n=1}^{N} \lambda_n y_n = 0 \tag{5}$$

$$\lambda_n = C - \mu_n \quad \text{với mọi} \quad n = 1, 2, \ldots, N \tag{6}$$

KKT system of Soft Margin SVM Opt. prob.

Nhận xét

- Nếu $\lambda_n = 0$, từ (6) và (13) $\Rightarrow \mu_n = C > 0$, $\xi_n = 0$, tức x_n thuộc vùng an toàn.
- Nếu $\lambda_n > 0$, kết hợp (12) suy ra

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

Có hai khả năng

- Nếu $0 < \lambda_n < C$ từ (6) $\Rightarrow \mu_n = C - \lambda_n > 0$ do đó $\xi_n = 0$ theo (13), tức là $y_n(\mathbf{w}^T\mathbf{x}_n + w_0) = 1$, x_n nằm ngay trên siêu phẳng lề (margin).
- Nếu $\lambda_n = C$ từ (6) $\Rightarrow \mu_n = 0$ do đó $\xi_n > 0$ nhận giá trị tùy ý. Lúc đó
  - Nếu $0 < \xi_n < 1$, x_n nằm trong vùng không an toàn nhưng vẫn được phân lớp đúng (đúng phía).
  - Ngược lại nếu $\xi_n > 1$, x_n bị phân lớp sai.

KKT system of Soft Margin SVM Opt. prob.

- Với bài toán trên, các điểm x_n ứng với $0 < \lambda_n < C$ sẽ được coi là support vectors, vì ta cần chúng để tìm $\mathbf{w}$ theo (4).
- Ta cần tập $\mathcal{M} = \{m : 0 < \lambda_m < C\}$ để tính w_0 và tập $\mathcal{S} = \{n : 0 < \lambda_n \leq C\}$ để tính $\mathbf{w}$ theo công thức

$$\mathbf{w} = \sum_{n \in \mathcal{S}} \lambda_n y_n \mathbf{x}_n \tag{14}$$

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

KKT system of Soft Margin SVM Opt. prob.

- Chú ý mục tiêu cuối cùng của phương pháp là phân loại (xác định nhãn y) cho điểm dữ liệu $\mathbf{x}$, chứ không phải tính $\mathbf{w}$, w_0.
- Do theo cách xây dựng $y = \text{sign}(\mathbf{w}^T\mathbf{x} + w_0)$ nên ta quan tâm đến giá trị

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

- Từ công thức trên ta thấy để phân loại dữ liệu $\mathbf{x}$, ta chỉ cần tính các λ_n, sau đó sử dụng các phép lấy tích vô hướng trên dữ liệu đã biết.

Unconstrained SoftMargin SVM Optimization prob.

- Ta tìm cách đưa bài toán tối ưu có ràng buộc (2) về dạng bài toán không ràng buộc tương đương để có thể giải bằng các phương pháp quen thuộc - ví dụ Gradient Descent.
- Điều kiện ràng buộc trên ξ_n: Với mọi $n = 1, 2, \ldots, N$, $\xi_n \geq 0$ và

$$1 - \xi_n - y_n(\mathbf{w}^T\mathbf{x} + w_0)) \leq 0 \Leftrightarrow \xi_n \geq 1 - y_n(\mathbf{w}^T\mathbf{x} + w_0))$$

  kết hợp lại thu được điều kiện tương đương $\xi_n \geq \max(0, 1 - y_n(\mathbf{w}^T\mathbf{x} + w_0))$.
- Do vậy, bài toán tối ưu có ràng buộc (2) có thể viết dưới dạng

$$(\mathbf{w}, w_0, \xi) = \arg\min_{\mathbf{w}, w_0, \xi} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{n=1}^{N} \xi_n$$
$$\text{subject to:} \quad \xi_n \geq \max(0, 1 - y_n(\mathbf{w}^T\mathbf{x} + w_0)), \ \forall n = 1, 2, \ldots, N \qquad (16)$$

Unconstrained SoftMargin SVM Optimization prob.

Ta xét tính chất sau:

Bổ đề 1

Giả sử $(\mathbf{w}, w_0, \xi)$ *là nghiệm của bài toán tối ưu* (16)*, tức là tại đó hàm mục tiêu đạt giá trị nhỏ nhất, thì:*

$$\xi_n = \max(0, 1 - y_n(\mathbf{w}^T \mathbf{x}_n + w_0)), \ \forall n = 1, 2, \ldots, N \tag{17}$$

Unconstrained SoftMargin SVM Optimization prob.

Chứng minh. Thật vậy, giả sử ngược lại, tồn tại n sao cho

$$\xi_n > \max(0, 1 - y_n(\mathbf{w}^T\mathbf{x}_n + w_0)).$$

Ta chọn $\xi'_n = \max(0, 1 - y_n(\mathbf{w}^T\mathbf{x}_n + w_0))$, lúc đó tất cả ràng buộc của bài toán vẫn thỏa mãn, và giá trị hàm mục tiêu sẽ là

$$\frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{l=1, l\neq n}^{N} \xi_l + \xi'_n < \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{n=1}^{N} \xi_n$$

tức là hàm mục tiêu có thể đạt giá trị nhỏ hơn. Điều này mâu thuẫn với giả thiết $(\mathbf{w}, w_0, \xi)$ là nghiệm của bài toán tối ưu (16).
Từ đó ta có điều cần chứng minh.

Unconstrained SoftMargin SVM Optimization prob.

- Ta thay toàn bộ các giá trị của ξ_n bởi vế phải trong (17) và thu được bài toán tương đương

$$(\mathbf{w}, b, \xi) = \arg \min_{\mathbf{w}, w_0, \xi} \frac{1}{2}\|\mathbf{w}\|_2^2 + C \sum_{n=1}^{N} \max(0, 1 - y_n(\mathbf{w}^T \mathbf{x}_n + w_0)) \quad (18)$$

$$\text{subject to: } \xi_n = \max(0, 1 - y_n(\mathbf{w}^T \mathbf{x}_n + w_0)), \ \forall n = 1, 2, \ldots, N$$

- Rõ ràng ξ_n không còn vai trò nữa, ta có thể lược bỏ nó mà không làm thay đổi nghiệm của bài toán và thu được bài toán tương đương

$$\begin{aligned}(\mathbf{w}, w_0) &= \arg \min_{\mathbf{w}, w_0} \frac{1}{2}\|\mathbf{w}\|_2^2 + C \sum_{n=1}^{N} \max(0, 1 - y_n(\mathbf{w}^T \mathbf{x}_n + w_0)) \\ &\triangleq \arg \min_{\mathbf{w}, w_0} J(\mathbf{w}, w_0).\end{aligned} \quad (19)$$

Unconstrained SoftMargin SVM Optimization prob.

- Nhận xét về (19):
  - Là bài toán tối ưu không ràng buộc
  - Tương đương với bài toán quy hoạch toàn phương (16), do đó có nghiệm toàn cục
  - Có thể lấy đạo hàm hàm mục tiêu $J(\mathbf{w}, w_0)$ theo $\mathbf{w}, w_0$
- Vậy có thể giải được (19) bằng các phương pháp như Gradient Descent.

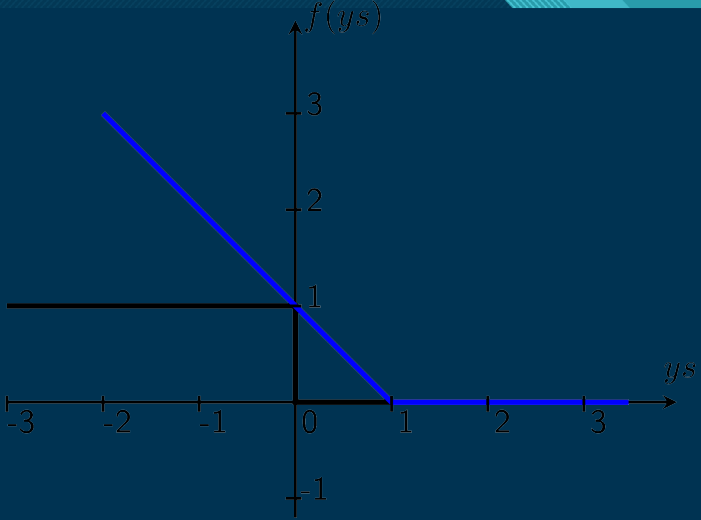
Hinge loss function

- Cách xây dựng phần phạt trong hàm mất mát như trên còn gọi là hàm Hinge loss.
- Hàm số này cũng được sử dụng nhiều trong các mô hình phân loại. Xét cho một cặp dữ liệu (x_n, y_n) và bộ tham số $(\mathbf{w}, w_0)$

$$J_n(\mathbf{w}, w_0) = \max(0, 1 - y_n z_n) \quad \text{trong đó} \quad z_n = \mathbf{w}x_n + w_0.$$

- Trong hàm *hinge loss,* z_n *được coi là score (điểm đánh giá)* x_n.
- Chúng ta gọi hàm đếm các điểm bị phân loại sai với bộ $(\mathbf{w}, w_0)$ (misclassified) là *Hàm zero-one loss.*
- Hình ở trang sau minh họa sự khác nhau giữa Hinge Loss và Zero-one Loss.

Hinge-Loss vs. Zero-one Loss



Hinge loss function

- Trong hình trên, yz là tích của đầu ra đúng (ground truth) y và đầu ra tính được (score) $z = \mathbf{w}^T x + w_0$. Phía phải của trục tung ứng với những điểm được phân loại đúng: $y \equiv sign(z)$.
- Phía trái của trục tung ứng với các điểm bị phân loại sai: $y = -sign(z)$.
- Với zero-one loss:
  - Các điểm có $y = -sign(z)$ gây ra mất mát như nhau ($f(yz) \equiv 1$), bất kể chúng ở gần hay xa đường phân chia (trục tung).
  - Là hàm rời rạc, rất khó tối ưu và khó định lượng đúng lượng phạt cần thiết như đã định nghĩa ở phần đầu.

Hinge loss function

- Với Hinge loss:
  - Những điểm ứng với $yz \geq 1$ (vùng an toàn), có $f(yz) \equiv 0$.
  - Những điểm nằm giữa margin của class tương ứng và siêu phẳng phân chia với có $0 < yz < 1$ gây ra một lượng phạt nhỏ.
  - Những điểm bị misclassifed, tức $yz < 0$ gây ra lượng phạt lớn hơn.
  - Khi tối thiểu hàm mất mát, ta cần tránh được những điểm bị misclassifed và lấn sang phần class còn lại quá nhiều. Đây chính là một ưu điểm của hàm hinge loss.
  - Là hàm liên tục, và có đạo hàm hầu khắp nơi (almost everywhere differentiable) trừ điểm có hoành độ bằng 1. Đạo hàm của Hinge loss dễ xác định (giống như hàm ReLU).

Loss-function from Hinge-loss view

- Phần trước, ta xây dựng hàm tổn thất với phần gốc $\|\mathbf{w}\|_2^2$ xuất phát từ cực đại hóa margin (như hard margin). Lượng phạt là phần bổ sung.
- Dưới đây, ta sẽ xây dựng hàm tổn thất xuất phát từ phần lượng phạt - Hinge-loss function, và $\|\mathbf{w}\|_2^2$ trở thành phần bổ sung.
  - Với mỗi cặp dữ liệu $(\mathbf{x}_n, y_n)$ và bộ tham số $(\mathbf{w}, w_0)$, đặt lượng tổn thất

$$L_n(\mathbf{w}, w_0) = \max(0, 1 - y_n(\mathbf{w}^T\mathbf{x}_n + w_0)).$$

  - Lấy tổng tất cả lượng tổn thất trên theo n (cho N cặp dữ liệu), ta được tổng lượng tổn thất cho N cặp điểm

$$L(\mathbf{w}, w_0) = \sum_{n=1}^{N} L_n = \sum_{n=1}^{N} \max(0, 1 - y_n(\mathbf{w}^T\mathbf{x}_n + w_0))$$

  - Nếu trực tiếp tối ưu tổng các hinge-loss $L(\mathbf{w}, w_0)$ sẽ thu được kết quả gì?

Loss-function from Hinge-loss view

- Nếu tập dữ liệu $(\mathbf{x}_n, y_n)_{n=1}^N$ là tách được tuyến tính, thì bộ tham số $(\mathbf{w}, w_0)$ tối ưu cần đảm bảo tổng tổn thất $L(\mathbf{w}, w_0) = 0$. Điều này đồng nghĩa với

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

- Dễ thấy với mọi hệ số $a > 0$, bộ tham số $(a\mathbf{w}, aw_0)$ cũng là nghiệm tối ưu vì

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

và do đó $L(a\mathbf{w}, aw_0) = 0$.

Loss-function from Hinge-loss view

- Phân tích trên cho thấy nếu chỉ dùng Hinge-Loss để xây dựng hàm tổn thất, ta sẽ có vô số nghiệm tối ưu với giá trị lớn tùy ý!
- Để tránh hiện tượng này, chúng ta bổ sung lượng hiệu chỉnh $R(\mathbf{w}, w_0)$ vào: $J(\mathbf{w}, w_0) = L(\mathbf{w}, w_0) + \lambda R(\mathbf{w}, w_0)$.
- Ở đây ta sẽ dùng hiệu chỉnh L_2 và thu được hàm tổn thất

$$\begin{aligned} J(\mathbf{w}, w_0) &= L(\mathbf{w}, w_0) + \lambda R(\mathbf{w}, w_0) \\ &= \sum_{n=1}^{N} \max(0, 1 - y_n(\mathbf{w}^T \mathbf{x}_n + w_0)) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2. \end{aligned} \tag{20}$$

- Tham số hiệu chỉnh $\lambda > 0$ và tương tự các mô hình trước, phần hiệu chỉnh chỉ cần tác động lên $\mathbf{w}$, còn gọi là kỹ thuật hiệu chỉnh *weight decay*.
- Đặt $C := 2/\lambda$, ta có (19) tương đương với (20).

Loss-function from Hinge-loss view

- Mở rộng các điểm dữ liệu $\mathbf{x}_n$ bằng cách thêm tọa độ $x_{n,0}$ và thu được $\bar{\mathbf{x}}_n \in \mathbb{R}^{d+1}$. Ghép w_0 thành tọa độ của $\mathbf{w}$ và thu được $\bar{\mathbf{w}} \in \mathbb{R}^{d+1}$, ta có thể viết gọn (20)

$$J(\bar{\mathbf{w}}) = \underbrace{\sum_{n=1}^{N} \max(0, 1 - y_n \bar{\mathbf{w}}^T \bar{\mathbf{x}}_n)}_{\text{hinge loss}} + \underbrace{\frac{\lambda}{2}\|\mathbf{w}\|_2^2}_{\text{regularization}}$$

- Có thể thấy $J(\bar{\mathbf{w}})$ là một hàm lồi theo $\bar{\mathbf{w}}$ vì
  - Hàm max là hàm lồi; hàm hằng (0) là hàm lồi và phần tuyến tính $(1 - y_n \bar{\mathbf{w}}^T \bar{\mathbf{x}}_n)$ lồi theo $\bar{\mathbf{w}}$, do đó phần hinge-loss là hàm lồi.
  - $\|\cdot\|_2$ là một hàm lồi, vậy số hạng $\frac{\lambda}{2}\|\mathbf{w}\|_2^2$ cũng là một hàm lồi.
  - Tổng của hai hàm lồi là một hàm lồi, do đó $J(\bar{\mathbf{w}})$ là hàm lồi.

Loss-function from Hinge-loss view

- Do (20) là hàm lồi, nên bài toán tối ưu tương ứng có nghiệm toàn cục, và có thể giải bằng phương pháp Gradient Descent:

$$\bar{\mathbf{w}}_{new} = \bar{\mathbf{w}}_{old} - \eta \nabla_{\bar{\mathbf{w}}} J(\bar{\mathbf{w}}_{old})$$

với $\eta > 0$ là hệ số học. Ta cần tính đạo hàm

$$\nabla_{\bar{\mathbf{w}}} J(\bar{\mathbf{w}}) = \nabla_{\bar{\mathbf{w}}} \left[\sum_{n=1}^{N} \max(0, 1 - y_n \bar{\mathbf{w}}^T \bar{\mathbf{x}}_n) \right] + \nabla_{\bar{\mathbf{w}}} \left[\frac{\lambda}{2} \|\mathbf{w}\|_2^2 \right].$$

- Đạo hàm phần hiệu chỉnh $\nabla_{\bar{\mathbf{w}}} \left\{ \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \right\}$ tính được dễ dàng nên ta sẽ chỉ xét chi tiết đạo hàm phần Hinge-Loss $\nabla_{\bar{\mathbf{w}}} \left[\sum_{n=1}^{N} \max(0, 1 - y_n \bar{\mathbf{w}}^T \bar{\mathbf{x}}_n) \right]$.

Minimizing Loss-Function

- Tính đạo hàm phần Hinge-loss $\nabla_{\bar{\mathbf{w}}}\left[\sum_{n=1}^{N}\max(0, 1 - y_n\bar{\mathbf{w}}^T\bar{\mathbf{x}}_n)\right]$:
- Xét tại từng điểm dữ liệu $(\mathbf{x}_n, y_n)$ ta có hai trường hợp
  - Nếu $1 - y_n\bar{\mathbf{w}}^T\bar{\mathbf{x}}_n \leq 0$ ta có $\nabla_{\bar{\mathbf{w}}}\left[\max(0, 1 - y_n\bar{\mathbf{w}}^T\bar{\mathbf{x}}_n)\right] = 0$.
  - Nếu $1 - y_n\bar{\mathbf{w}}^T\bar{\mathbf{x}}_n > 0$ ta có $\nabla_{\bar{\mathbf{w}}}\left[\max(0, 1 - y_n\bar{\mathbf{w}}^T\bar{\mathbf{x}}_n)\right] = -y_n\mathbf{x}_n$.
- Đặt

$$\mathbf{Z} = [y_1\bar{\mathbf{x}}_1, y_2\bar{\mathbf{x}}_2, \ldots, y_N\bar{\mathbf{x}}_N] \tag{21}$$

$$\mathbf{u} = [y_1\bar{\mathbf{w}}^T\bar{\mathbf{x}}_1, y_2\bar{\mathbf{w}}^T\bar{\mathbf{x}}_2, \ldots, y_N\bar{\mathbf{w}}^T\bar{\mathbf{x}}_N] = \bar{\mathbf{w}}^T\mathbf{Z} \tag{22}$$

chú ý $\mathbf{u} \in \mathbb{R}^{1\times N}$ - tức là vector dòng N phần tử. Xét tập các chỉ số

$$\mathcal{H} = \{n : u_n < 1\} \quad \text{- tất cả các tọa độ } n \text{ có } \quad \mathbf{u}_n < 1.$$

Minimizing Loss-Function

- Tổng hợp lại, ta có đạo hàm hàm tổn thất theo $\bar{\mathbf{w}}$:

$$\nabla J(\bar{\mathbf{w}}) = \sum_{n \in \mathcal{H}} -y_n \bar{\mathbf{x}}_n + \lambda \begin{bmatrix} 0 \\ \mathbf{w} \end{bmatrix} \tag{23}$$

- Vậy phương pháp lặp Gradient Descent trở thành

$$\bar{\mathbf{w}} = \bar{\mathbf{w}} - \eta \left(\sum_{n \in \mathcal{H}} -y_n \bar{\mathbf{x}}_n + \lambda \begin{bmatrix} 0 \\ \mathbf{w} \end{bmatrix} \right) \tag{24}$$

với $\eta > 0$ là hệ số học. Đến đây ta có thể sử dụng phương pháp Mini-Batch Gradient Descent để giải.