Support Vector Machines and Kernel Methods

Ju Sun*

November 18, 2024

Overview The support vector machine (SVM) was invented by Vladimir N. Vapnik around mid 90's, rendering neural networks out of flavor in machine learning for 15 years until 2010's—when neural networks stroke back and took the main arena of machine learning until now. SVMs and kernel methods in general are powerful and effective in practice and well grounded in theory; they remain competitive or even superior on structured data, compared to alternatives.

We first rederive the hard-margin SVM from the geometric view of margin maximization. Then, using tools from convex analysis, we show the key properties of hard-margin SVMs. After that, we formulate and analyze soft-margin SVMs, and also discuss scalable optimization methods for SVM training. Afterward we move to kernel methods, which really help SVMs and other linear models take off and enable easy and practical nonlinear modeling. We wrap up the chapter by briefly introducing kernel methods for learning settings beyond binary classification.

Contents

*Department of Computer Science and Engineering, University of Minnesota at Twin Cities. Email: jusun@umn.edu.

1 Hard-margin SVMs

Our starting point is a *linearly separable* training set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{1, -1\}$ for all $i \in [N]$, for binary classification. Here, linear separability means that there exists a pair $(\boldsymbol{w}_0, b_0)$ so that $\operatorname{sign}(\langle \boldsymbol{w}_0, \boldsymbol{x}_i \rangle + b_0) = y_i \iff y_i(\langle \boldsymbol{w}_0, \boldsymbol{x}_i \rangle + b_0) > 0$ for all $i \in [N]$. In other words, the positive

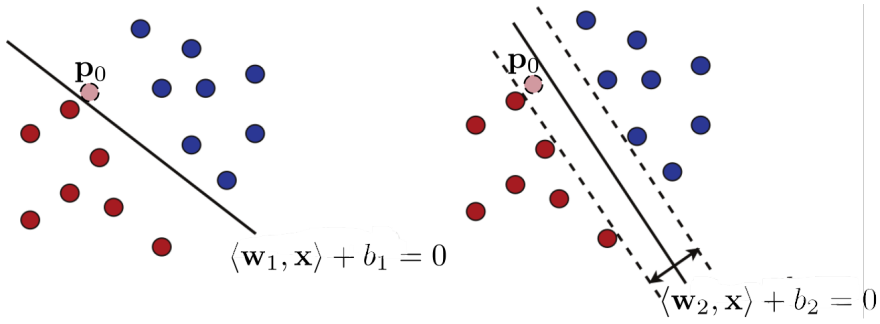


Figure 1: (left & right) A linearly separable dataset (ignoring $\boldsymbol{p}_0$ which is a test point) can be separated by infinitely many hyperplanes. (right) Hard-margin SVM finds the separating hyperplane that leaves the largest margins on the positive and negative sides. Adapted from Figure 5.1 of [Moh18].

and negative points can be perfectly separated by a hyperplane[1].

But is there a unique separating hyperplane? From Fig. 1, it is easy to see that the answer is no. Here, ignoring $\boldsymbol{p}_0$ which is a test point, we can find two separating hyperplanes $H_1 = \{\boldsymbol{x} : \langle \boldsymbol{w}_1, \boldsymbol{x} \rangle + b_1 = 0\}$ and $H_2 = \{\boldsymbol{x} : \langle \boldsymbol{w}_2, \boldsymbol{x} \rangle + b_2 = 0\}$; both perfectly separate the training set. In fact, one can construct infinitely many by slightly rotating and shifting H_1 and H_2.

Which separating hyperplanes are better than others? There are many possible answers. A sensible criterion is *robust classification*: ideally, points that deviate slightly from the positive points should be classified as positive and similarly for the negative side. In Fig. 1, the test point $\boldsymbol{p}_0$ is closest to (in ℓ_2 distance) a red training point, but H_1 labels it as "blue" whereas H_2 labels it as "red". Obviously, H_1 is not robust as it is very close to some red (and also blue) points. In contrast, H_2 is relatively robust, as it maintains reasonable distances to all training points. We will formalize the intuition below.

[1] Recall that since linear hyperplanes are also affine hyperplanes, we refer to affine hyperplanes by default when we say hyperplanes.

1.1 The max-margin principle

First, we need to define the distance of a point to a hyperplane—which is a set of points. For simplicity, we focus on the point-hyperplane distance induced by the ℓ_2 distance, although, in principle, the distance can be induced by any distance.

Lemma 1.1 ((ℓ_2) Geometric margin)**.** *For any point $\boldsymbol{x}_0 \in \mathbb{R}^d$ and any hyperplane $H = \{\boldsymbol{x} \in \mathbb{R}^d : \boldsymbol{w}^\intercal \boldsymbol{x} + b = 0\}$, the ℓ_2 distance, or ℓ_2 **geometric margin**, between $\boldsymbol{x}_0$ and H is: $d_{\ell_2}(\boldsymbol{x}_0, H) = |\boldsymbol{w}^\intercal \boldsymbol{x}_0 + b|/\|\boldsymbol{w}\|_2$.*

Proof. See Appendix A. ■

Recall that to achieve robust classification, we hope to keep all training points away from the separating hyperplane H. A natural objective is to make all the geometric margins as large as possible, or equivalently the worst geometric margin as large as possible, i.e.,

$$\max_{\boldsymbol{w},b} \min_{i\in[N]} \frac{|\boldsymbol{w}^\intercal \boldsymbol{x}_i + b|}{\|\boldsymbol{w}\|_2} \quad \text{s.t. } y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) > 0 \; \forall \, i \in [N]. \tag{1.1}$$

This implements the *max-margin principle*, i.e., to maximize the worst geometric margin.

1.2 Derivation of hard-margin SVMs

There are several numerical issues when solving Eq. (1.1) directly, which is nonconvex and nonsmooth. Below, we will reformulate Eq. (1.1) into an equivalent convex form, which is much more tractable numerically.

First, since $|\boldsymbol{w}^\intercal \boldsymbol{x}_i + b| = y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b)$ when $y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) > 0$ for all i, so our first simplified version is:

$$\max_{\boldsymbol{w},b} \min_{i\in[N]} \frac{y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b)}{\|\boldsymbol{w}\|_2} \quad \text{s.t. } y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) > 0 \; \forall \, i \in [N], \tag{1.2}$$

which removes the nonsmoothness due to the $|\cdot|$ function.

Next, observe that for any global optimizer $(\boldsymbol{w}_*, b_*)$, we have that $\lambda(\boldsymbol{w}_*, b_*)$ is also a global optimizer for all $\lambda > 0$, as

$$y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) > 0 \; \forall \, i \in [N] \iff y_i \lambda(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) > 0 \quad \forall \, i \in [N], \; \forall \lambda > 0, \tag{1.3}$$

$$\text{and} \quad (y_i \lambda(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b))/\|\lambda \boldsymbol{w}\|_2 = (y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b))/\|\boldsymbol{w}\|_2 \quad \forall \lambda > 0. \tag{1.4}$$

This implies that, without loss of generality, we can seek a global optimizer so that

$$\min_{i\in[N]} y_i(\boldsymbol{w}^\intercal x_i + b) = 1, \tag{1.5}$$

turning Eq. (1.2) into

$$\max_{\boldsymbol{w},b} \min_{i\in[N]} \frac{y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b)}{\|\boldsymbol{w}\|_2} \quad \text{s.t. } \min_{i\in[N]} y_i(\boldsymbol{w}^\intercal x_i + b) = 1, \tag{1.6}$$

or

$$\max_{\boldsymbol{w},b} \frac{1}{\|\boldsymbol{w}\|_2} \quad \text{s.t. } \min_{i\in[N]} y_i(\boldsymbol{w}^\intercal x_i + b) = 1, \tag{1.7}$$

after we substitute the constraint into the objective, or

$$\min_{\boldsymbol{w},b} \|\boldsymbol{w}\|_2 \quad \text{s.t. } \min_{i\in[N]} y_i(\boldsymbol{w}^\intercal x_i + b) = 1, \tag{1.8}$$

as $\max\ 1/\|\boldsymbol{w}\|_2$ is equivalent to $\min\ \|\boldsymbol{w}\|_2$.

Now the objective looks neat, but the constraint is a bit awful—the pointwise min operation typically induces nonsmooth points. Now we claim that Eq. (1.8) is equivalent to

$$\min_{\boldsymbol{w},b}\ \|\boldsymbol{w}\|_2 \quad \text{s.t. } y_i(\boldsymbol{w}^\intercal x_i + b) \geq 1\ \forall\ i \in [N]. \tag{1.9}$$

Indeed, any minimizer $(\boldsymbol{w}_*, b_*)$ to problem (1.9) must satisfy $\min_{i\in[N]}\ y_i(\boldsymbol{w}_*^\intercal \boldsymbol{x}_i + b_*) = 1$. Suppose not, then $\min_{i\in[N]}\ y_i(\langle \boldsymbol{w}_*, \boldsymbol{x}_i\rangle + b_*) = 1 + \varepsilon$ for a certain $\varepsilon > 0$. Now $\frac{1}{1+\varepsilon}(\boldsymbol{w}_*, b_*)$ is a feasible point with a strictly lower objective value $\frac{1}{1+\varepsilon}\|\boldsymbol{w}_*\|_2 < \|\boldsymbol{w}_*\|_2$, rendering $(\boldsymbol{w}_*, b_*)$ a non-minimizer—a contradiction. This implies that $(\boldsymbol{w}_*, b_*)$ is feasible for problem Eq. (1.8). However, since the constraint set of Eq. (1.8) is a strict subset of that of Eq. (1.9), the optimal value achieved in Eq. (1.8) should not be less than that achieved in Eq. (1.9). Now that $(\boldsymbol{w}_*, b_*)$ is a minimizer of Eq. (1.9) (i.e., achieving the optimal value) but also feasible for Eq. (1.8), it is also a minimizer to Eq. (1.8).

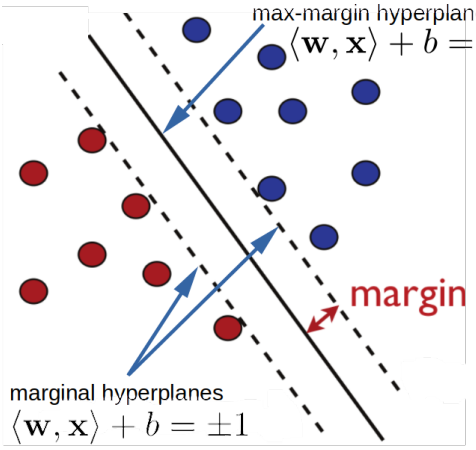


Figure 2: Illustration of the hard-margin SVM, and the associated max-margin and marginal hyperplanes (figure adapted from Figure 5.3 of [Moh18]).

After applying a slight aesthetic tweak to the objective—$\|\boldsymbol{w}\|_2$ into $\frac{1}{2}\|\boldsymbol{w}\|_2^2$ to make the objective everywhere differentiable, we obtain the famous *hard-margin SVM* formulation:

$$\boxed{\min_{\boldsymbol{w},b}\ \frac{1}{2}\|\boldsymbol{w}\|_2^2 \quad \text{s.t. } y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) \geq 1\ \forall\ i \in [N].} \tag{1.10}$$

Back to our geometric picture, the hard-margin SVM tries to find a hyperplane that separates the two classes and leaves the largest geometric margins for the training set. We call such a hyperplane a *max-margin hyperplane*. Parallel to the max-margin hyperplane, we define two hyperplanes that pass through the nearest point(s) to the max-margin hyperplane on the positive and negative sides, respectively. They are called *marginal hyperplanes*.

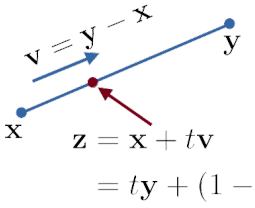
Several observations can be made (see Fig. 2): **(1) Robust hyperplanes**. If we slightly perturb "interior" points that are away from the marginal hyperplane on each side or add similar interior points, the max-margin hyperplane and also the marginal hyperplanes will not change for the altered dataset. In other words, these hyperplanes are only determined by the few points on the "frontiers" that support the marginal hyperplanes. These points are called *support vectors*, as we shall see in Section 3; **(2) Robust classification**: all training points are at least $1/\|\boldsymbol{w}_*\|_2$ away in ℓ_2 distance from the decision boundary—the max-margin hyperplane. So, perturbations no larger than $1/\|\boldsymbol{w}_*\|_2$ in ℓ_2 norm will not cause classification errors; **(3) Equal margins**: The max-margin hyperplane will leave equal margins on both sides. In fact, if the margins were not equal, say the positive side is smaller, one can move the hyperplane parallel to the normal direction $\boldsymbol{w}$ to make them equal, which at the same time improves the worst geometric margin.

To formalize these observations, we need tools from convex analysis. Moreover, these tools can help us answer other questions that are not clear based on geometric observations alone, e.g., whether the max-margin hyperplane is unique.

2 Quick review of convex analysis

2.1 Convex sets

The line segment connecting two points $\boldsymbol{x}, \boldsymbol{y}$ can be represented as follows. Starting from $\boldsymbol{x}$, any point $\boldsymbol{z}$ on the segment can be written as $\boldsymbol{z} = \boldsymbol{x} + t\boldsymbol{v}$ where $\boldsymbol{v} = \boldsymbol{y} - \boldsymbol{x}$ and $t \in [0, 1]$. Since $\boldsymbol{x} + t\boldsymbol{v} = t\boldsymbol{y} + (1-t)\boldsymbol{x}$, the line segment can be written as



$$\{t\boldsymbol{y} + (1-t)\boldsymbol{x} : t \in [0,1]\} = \{t\boldsymbol{x} + (1-t)\boldsymbol{y} : t \in [0,1]\}\,. \tag{2.1}$$

A set S is said to be convex if every line segment connecting every two points of the set stays in the set, i.e.,

$$\{t\boldsymbol{x} + (1-t)\boldsymbol{y} : t \in [0,1]\} \subset S \quad \forall\, \boldsymbol{x}, \boldsymbol{y} \in S. \tag{2.2}$$

Figure 3 (left) show numerous examples of convex and nonconvex sets.

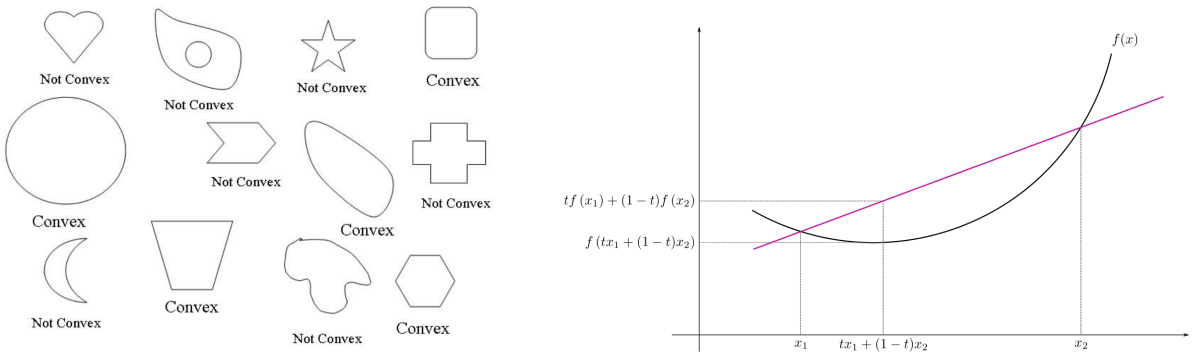


Figure 3: Illustration of convex sets (left; image credit: http://www2.econ.iastate.edu/classes/econ500/hallam/documents/Convex_Opt_000.pdf) and convex functions (right; image credit: wikipedia).

When we face a complex set and try to tell it is convex, it is typically hard to do so from the definition. The following operation rules can become handy in these scenarios.

Theorem 2.1 (Operations that preserve convexity of sets)**.** *For any two convex sets $S_1, S_2 \subset \mathbb{R}^n$, the following sets are also convex:* **(1)** *set product $S_1 \times S_2 \doteq \{(\boldsymbol{x}_1, \boldsymbol{x}_2) : \boldsymbol{x}_1 \in S_1, \boldsymbol{x}_2 \in S_2\}$,* **(2)** ***set sum*** *$S_1 + S_2 = \{\boldsymbol{x}_1 + \boldsymbol{x}_2 : \boldsymbol{x}_1 \in S_1, \boldsymbol{x}_2 \in S_2\}$,* **(3)** ***set intersection*** *$S_1 \cap S_2$, and* **(4)** ***set projection*** *$\{(x_1, \ldots, x_k) : \boldsymbol{x} \in S_1\}$ for any $k \in [n]$.*

2.2 Convex functions

Let X be a convex set. A function $f : X \mapsto \mathbb{R}$ is said to be convex if every chord connecting any two distinct points on the graph of f lies above (i.e., not below) the graph, as illustrated in Fig. 3(right). Mathematically, this means

$$f(t\boldsymbol{x} + (1-t)\boldsymbol{y}) \le tf(\boldsymbol{x}) + (1-t)f(\boldsymbol{y}) \quad \forall\, \boldsymbol{x}, \boldsymbol{y} \in X \text{ and } \forall\, t \in [0,1]. \tag{2.3}$$

To verify a function is convex, it is important to check **(1)** the domain X is a convex set, and **(2)** f satisfies the condition in Eq. (2.3).

Examples of convex functions:

- **All vector and matrix norms**. Any vector norm $\|\cdot\|$ satisfies the triangle inequality $\|\boldsymbol{u}+\boldsymbol{v}\| \leq \|\boldsymbol{u}\|+\|\boldsymbol{v}\|$ by the definition of norms, so we have

$$\|t\boldsymbol{u}+(1-t)\boldsymbol{v}\| \leq \|t\boldsymbol{u}\| + \|(1-t)\boldsymbol{v}\| = t\|\boldsymbol{u}\| + (1-t)\|\boldsymbol{v}\| \quad \forall\, \boldsymbol{u},\boldsymbol{v}\ \forall\, t \in [0,1], \tag{2.4}$$

  verifying the definition of convex functions. Similar argument can be carried out for all matrix norms.

- All linear functions of the form $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}$. This can be easily verified from the definition.

The most salient property of convex functions is that **any local minimizer of a convex function is also a global minimizer**.

In the definition of convex functions, when every chord connecting any two distinct points lies *strictly* above the the graph except for the two end points, the function is called *strictly convex*, i.e.,

$$f(t\boldsymbol{x}+(1-t)\boldsymbol{y}) < tf(\boldsymbol{x})+(1-t)f(\boldsymbol{y}) \quad \forall\, \boldsymbol{x},\boldsymbol{y} \in X \text{ and } \forall\, t \in (0,1). \tag{2.5}$$

Any strictly convex function has a unique global minimizer.

Let X be a convex set and consider a first-order differentiable function $f : X \mapsto \mathbb{R}$. We have the following equivalent properties:

$$f \text{ is convex} \iff f(\boldsymbol{y}) - f(\boldsymbol{x}) \geq \langle \nabla f(\boldsymbol{x}), \boldsymbol{y}-\boldsymbol{x}\rangle \quad \forall\, \boldsymbol{x},\boldsymbol{y} \in X \tag{2.6}$$

$$f \text{ is strictly convex} \iff f(\boldsymbol{y}) - f(\boldsymbol{x}) > \langle \nabla f(\boldsymbol{x}), \boldsymbol{y}-\boldsymbol{x}\rangle \quad \forall \text{ distinct } \boldsymbol{x},\boldsymbol{y} \in X \tag{2.7}$$

If f is second-order differentiable, we have

$$f \text{ is convex} \iff \nabla^2 g(\boldsymbol{x}) \succeq \boldsymbol{0} \quad \forall\, \boldsymbol{x} \in X \tag{2.8}$$

$$f \text{ is strictly convex} \impliedby \nabla^2 g(\boldsymbol{x}) \succ \boldsymbol{0} \quad \forall\, \boldsymbol{x} \in X, \tag{2.9}$$

where note that the last one is one-directional.

For constrained optimization problems, we often see constraints of the form $f(\boldsymbol{x}) \leq 0$. There is a very useful result that connects the convexity of f to the convexity of the constraint set: if $f(\boldsymbol{x}) : X \mapsto \mathbb{R}$ is a convex function, the **sublevel set** $\{\boldsymbol{x} \in X : f(\boldsymbol{x}) \leq 0\}$ is a convex set.

Similar to the case for convex sets, there are operations that preserve the convexity of functions.

Theorem 2.2 (Operations that preserve convexity of functions)**.** *We have the following results.*
(1) ***positive combinations of convex functions preserve convexity****: For convex functions $f_i : X \mapsto \mathbb{R}$ $(i = 1,\dots,K)$, $\sum_{i\in[K]} \alpha_i f_i$ is convex over X for all $\alpha_i \geq 0$;*
(2) ***pointwise maximum of convex functions preserve convexity****: For convex functions $f_i : X \mapsto \mathbb{R}$ $(i = 1,\dots,K)$, $\max_{i\in[K]} f_i$ is convex over X;*
(3) ***composition with linear functions preserve convexity****: if f is convex over the range of $\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}$ (i.e., $\{\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b} : \boldsymbol{x} \in X\}$), $f(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b})$ is convex over X;*
(4) ***composition of monotonic convex functions****: $h \circ g$ is convex over X if both $g : X \mapsto \mathbb{R}$ and $h : \mathbb{R} \mapsto \mathbb{R}$ are twice differentiable and either of the following holds:*
- *h is convex and non-decreasing and g is convex,*
- *h is convex and non-increasing and $-g$ is convex.*

(5) ***partial minimization of convex functions preserves convexity****: Let $f(\boldsymbol{x},\boldsymbol{y})$ be a convex function over $X \times Y$, where both X and Y are convex sets and hence $X \times Y$ is also a convex set. The partial minimization $\inf_{\boldsymbol{y}\in Y} f(\boldsymbol{x},\boldsymbol{y})$* [2] *is convex over X.*

[2] Recall that $\inf$ can be roughly treated as $\min$, but for $\min$ the minimum value must be achieved by a point inside the domain, whereas for $\inf$ we take a limit point that can be outside. An example is minimizing $f(x) = x^2$ over $(-\infty, 0)$. Here, $\min f(x)$ does not make sense, but $\inf f(x) = 0$. So, in general, it is safer to use $\inf$, instead of $\min$, when we mean to perform minimization.

2.3 Optimality conditions (KKT conditions) for constrained convex problems

The hard-margin SVM formulation in Eq. (1.10) is a constrained optimization problem. Hence, to characterize the properties of its solution, we need optimality conditions for constrained problems. Since the formulation is convex (formally argued in Section 3), we will focus only on such conditions for constrained convex problems.

Theorem 2.3 (Optimality conditions, or Karush-Kuhn-Tucker (KKT) conditions, for convex problems)**.** *Consider a convex optimization problem:*

$$\min_{\boldsymbol{x}} \; f(\boldsymbol{x}) \quad \text{s.t. } g_i(\boldsymbol{x}) \leq 0, \forall\, i \in \mathcal{I} \quad \textit{and} \quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} = \boldsymbol{0}, \tag{2.10}$$

where f and g_i's are continuously differentiable, convex functions and $\mathcal{I}$ is the index set for the inequality constraints. Define the Lagrangian function as

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\pi}, \boldsymbol{\lambda}) \doteq f(\boldsymbol{x}) + \sum\nolimits_{i\in\mathcal{I}} \pi_i g_i(\boldsymbol{x}) + \boldsymbol{\lambda}^\intercal(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}), \tag{2.11}$$

where $\boldsymbol{x}$ is called the primal variable, and $\boldsymbol{\pi} \geq \boldsymbol{0}$ and $\boldsymbol{\lambda}$ are called the dual variables. ***Suppose that the constraint set is* strictly feasible *(also called the* Slater's condition*), i.e., there exists an $\boldsymbol{x}_0$ so that $g_i(\boldsymbol{x}_0) < 0$ for all $i \in \mathcal{I}$ and $\boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{b} = \boldsymbol{0}$.*** *Then, $\boldsymbol{x}$ is a global minimizer is equivalent to the following:*

$$\exists \boldsymbol{\pi}, \boldsymbol{\lambda} \geq \boldsymbol{0} \text{ s.t. } \begin{cases} \textbf{\textit{stationarity}} : \partial_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\pi}, \boldsymbol{\lambda}) = \boldsymbol{0} \\ \textbf{\textit{feasibility}} : g_i(\boldsymbol{x}) \leq 0 \;\forall\, i \in \mathcal{I}, \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} = \boldsymbol{0}, \boldsymbol{\pi} \geq \boldsymbol{0} \\ \textbf{\textit{complementary slackness}} : \pi_i g_i(\boldsymbol{x}) = 0 \;\forall i \in \mathcal{I} \iff \sum_{i\in\mathcal{I}} \pi_i g_i(\boldsymbol{x}) = 0. \end{cases}$$

Note that to apply the KKT condition, it is crucial to check the Slater's condition first. Also, the simultaneous properties, including stationarity, feasibiliy, and complementary slackness, are sufficient and necessary for checking a global minimizer for a constrained convex problem.

3 Key properties of hard-margin SVMs

With all the essential tools of convex analysis, we are now ready to study the properties of the solution to the hard-margin SVM.

We start by checking that the hard-margin formulation in Eq. (1.10) is indeed a convex problem. First, for each $i \in [N]$, $1 - y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b)$ is a linear function in $(\boldsymbol{w}, b)$ and therefore convex, so $\{(\boldsymbol{w}, b) : 1 - y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) \leq 0\}$ is a convex set in the $(\boldsymbol{w}, b)$ space. The whole constraint set is the intersection of N such convex sets, i.e.,

$$\{(\boldsymbol{w}, b) : 1 - y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) \leq 0 \;\forall i \in [N]\} = \bigcap\nolimits_{i\in[N]} \{(\boldsymbol{w}, b) : 1 - y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) \leq 0\}, \tag{3.1}$$

so is convex by the set intersection rule in Theorem 2.1. Now for the objective, we know that $\|\boldsymbol{w}\|_2$ is convex as a vector norm. So $\frac{1}{2}\|\boldsymbol{w}\|_2^2$ is a composition of the monotonically increasing function $h(z) = \frac{1}{2}z^2$ over $[0, \infty)$ with the convex function $f(\boldsymbol{w}) = \|\boldsymbol{w}\|_2$, and is thus convex by the monotonic composition rule in Theorem 2.2. [3] So we conclude that the hard-margin SVM as formulated in Eq. (1.10) is a constrained convex problem.

Now, to apply the KKT conditions from Theorem 2.3, we verify the Slater's condition first. Since we assume linear separability of the training set, i.e., there exists $(\boldsymbol{w}_0, b_0)$ so that $y_i(\boldsymbol{w}_0^\intercal \boldsymbol{x}_i + b) >$

[3] Another way to show that $\boldsymbol{f}(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$ is to check the Hessian: $\nabla^2 f(\boldsymbol{w}) = \boldsymbol{I} \succeq \boldsymbol{0}$.

$0, \forall i \in [N]$, which implies that one can find a scaling factor η_0 so that $\eta_0(\boldsymbol{w}_0, b_0)$ is strictly feasible, i.e., $y_i(\eta_0 \boldsymbol{w}_0^\intercal \boldsymbol{x}_i + \eta_0 b) > 1, \forall i \in [N]$. So Slater's condition is verified.

Now we can apply the KKT conditions. The Lagrangian function is

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\pi}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \sum\nolimits_{i\in[N]} \pi_i[1 - y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b)], \tag{3.2}$$

where we do not have equality constraints here. For a candidate global minimizer $(\boldsymbol{w}, b)$ and the associated dual variable $\boldsymbol{\pi}$, we have the following from the KKT conditions:

$$\textbf{(stationarity)}\ \partial_{\boldsymbol{w},b}\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\pi}) = \boldsymbol{0} \Longrightarrow \boldsymbol{w} = \sum\nolimits_{i\in[N]} \pi_i y_i \boldsymbol{x}_i,\ \sum\nolimits_{i\in[N]} \pi_i y_i = 0, \tag{3.3}$$

$$\textbf{(feasibility)}\ 1 - y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) \le 0\ \forall i \in [N], \quad \boldsymbol{\pi} \ge \boldsymbol{0}, \tag{3.4}$$

$$\textbf{(comp. slackness)}\ \pi_i[1 - y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b)] = 0\ \forall i \in [N]. \tag{3.5}$$

From these conditions, we can observe the following:

- **support vectors and marginal hyperplanes**. From the stationarity condition, the normal vector $\boldsymbol{w}$ is a linear combination of $\boldsymbol{x}_i$'s. When $\pi_i > 0$ (recall that π_i is always nonnegative), the corresponding point $\boldsymbol{x}_i$ contributes to $\boldsymbol{w}$, and these points are called *support vectors*. From complementary slackness, when $\pi_i > 0$, $y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) = 1$, i.e., $\boldsymbol{w}^\intercal \boldsymbol{x}_i + b = +1$ or -1. These define the marginal hyperplanes (see Fig. 2), where no point should lie between them.
- **support vectors come from both sides**. The result $\sum_{i\in[N]} \pi_i y_i = 0$ tells us that the support vectors should not be only positive or negative points, as positive combinations of $+1$ or -1 alone cannot be 0.
- **equal margins**. Since we have both positive and negative support vectors, i.e., $y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) = 1$ for both $y_i = 1$ and $y_i = -1$, and all other cases with $1 - y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) \le 0$, the geometric margin of the support vectors to the separating hyperplane on both sides is $1/\|\boldsymbol{w}\|_2$.

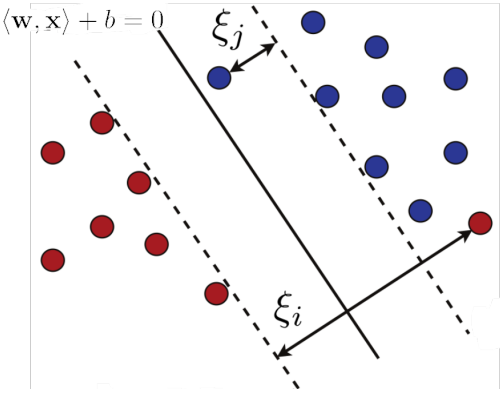


Figure 4: Illustration of the soft-margin SVM, and the associated max-margin and marginal hyperplanes (figure adapted from [Moh18]).

Moreover, through mathematical analysis, we can also show that the hard-margin SVM has a unique global minimizer, and adding and perturbing points away from the marginal hyperplanes will not change the separating hyperplanes (i.e., robust hyperplanes). We will do this in our homework.

4 Soft-margin SVMs and properties

A major issue with the hard-margin SVM is that it cannot deal with data that are not linearly separable: in these cases, there is no feasible point for Eq. (1.10). To work with non-separable data, we have to make compromises. One possibility is to allow classification errors while promoting large margins. This leads to the famous *soft-margin SVM*.

4.1 Soft-margin SVMs

The first step is to allow certain $\boldsymbol{x}_i$'s to move across marginal hyperplanes, i.e., by relaxing the constraints from $y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) \ge 1\ \forall\ i \in [N]$ to $y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) \ge 1 - \xi_i, \xi_i \ge 0\ \forall\ i \in [N]$, and hyperplane crossings occur for $\boldsymbol{x}_i$' s with $\xi_i > 0$. But we also do not want too many crossings, so it makes sense to penalize

the total crossings, e.g., by penalizing $\sum_{i\in[N]} \xi_i$ in the objective. Putting these together, we arrive at the standard *soft-margin SVM* (see Fig. 4):

$$\boxed{\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i\in[N]} \xi_i \quad \text{s.t. } y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0 \; \forall \; i \in [N]} \tag{4.1}$$

where $C > 0$ is a tradeoff parameter that controls the relative weight we put on the margin-crossing penalty term $\sum_{i\in[N]} \xi_i$. Note that the relaxation and penalization that we described above is not the only possibility—there are many versions of soft-margin SVMs.

The tradeoff parameter $C > 0$ in Eq. (4.1) balances margin maximization and hyperplane crossings. Recall that the geometric margin is $1/\|\boldsymbol{w}\|_2$. As C increases, the penalty for hyperplane crossings becomes more stressed, resulting in a smaller $\sum_{i\in[N]} \xi_i$, i.e., fewer crossings, and a larger $\|\boldsymbol{w}\|_2$, i.e., a smaller geometric margin.

Like for the hard-margin case, we can invoke the KKT conditions to derive quantitative results for the soft-margin case.

Since all the constraint functions are linear, the constraint set is convex. Moreover, the objective is a positive combination of the convex function $\frac{1}{2}\|\boldsymbol{w}\|_2^2$ and the (linear) convex function $\sum_{i\in[N]} \xi_i$, so it is convex. So, the soft-margin SVM formulated as in Eq. (4.1) is a constrained convex problem. To verify Slater's condition, it is easy to see that $\boldsymbol{w} = \boldsymbol{0}, b = 0, \boldsymbol{\xi} = 10 \times \boldsymbol{1}$ ($\boldsymbol{1}$ is an all-one vector) is a strictly feasible point.

4.2 Properties of soft-margin SVMs

Now we are ready to invoke the KKT conditions from Theorem 2.3. The Lagrangian function is

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\pi}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i\in[N]} \xi_i + \sum_{i\in[N]} \lambda_i(1 - \xi_i - y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b)) - \sum_{i\in[N]} \pi_i \xi_i. \tag{4.2}$$

The KKT conditions are

$$\text{stationarity}: \quad \boldsymbol{w} = \sum_{i\in[N]} \lambda_i y_i \boldsymbol{x}_i, \; \sum_{i\in[N]} \lambda_i y_i = 0, \; C\boldsymbol{1} = \boldsymbol{\lambda} + \boldsymbol{\pi} \tag{4.3}$$

$$\text{feasibility}: \quad y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) \geq 1 - \xi_i, \; \xi_i \geq 0, \; \lambda_i \geq 0, \; \pi_i \geq 0 \; \forall \; i \tag{4.4}$$

$$\text{complementary slackness}: \quad \lambda_i(1 - \xi_i - y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b)) = 0, \; \pi_i \xi_i = 0 \; \forall \; i \tag{4.5}$$

We can make several observations almost identical to the hard-margin case: $\boldsymbol{w}$ is a linear combination of $\boldsymbol{x}_i$'s, support vectors come from both sides, and the margin is equal on both sides (marginal hyperplanes are still of the form $\boldsymbol{w}^\intercal \boldsymbol{x}_i + b = \pm 1$). Now, more on the support vectors, i.e., when $\lambda_i > 0$ and so $y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) = 1 - \xi_i$. There are two cases:

- $\xi_i = 0$: so $y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) = 1$ and so the corresponding $\boldsymbol{x}_i$'s lie on one of the two marginal hyperplanes $\{\boldsymbol{x} : \boldsymbol{w}^\intercal \boldsymbol{x}_i + b = \pm 1\}$, as in the hard-margin case.
- $\xi_i > 0$: so $y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) = 1 - \xi_i$ and the corresponding $\boldsymbol{x}_i$'s are outliers—either correctly classified but with small margins (i.e., when $\xi_i \in (0, 1)$) or misclassified (i.e., when $\xi_i \in [1, \infty)$). In these cases, $\pi_i = 0$ as $\pi_i \xi_i = 0$, and by stationarity $\lambda_i = C$.

5 Optimizing SVMs

Now, we will focus on the computational issues around the standard soft-margin SVM as formulated in Eq. (4.1). We do not follow the classical ideas that solve soft-margin SVMs by solving dual

quadratic problems (e.g., as in the famous `LIBSVM` library [CL11] that is called by the `SVC` function in `scikit-learn`); instead, we solve the unconstrained reformulation directly, that, when coupled with stochastic optimization methods, tends to be more scalable—as implemented in the `SGDClassifier` function in `scikit-learn`.

5.1 Unconstrained reformulation of soft-margin SVMs

First, the constraints in Eq. (4.1) can be equivalently written as $\xi_i \geq \max(0, 1 - y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b))\ \forall i$, turning Eq. (4.1) into the equivalent form

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum\nolimits_{i\in[N]} \xi_i \quad \text{s.t. } \xi_i \geq \max(0, 1 - y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b))\ \forall\ i \in [N]. \tag{5.1}$$

Next, we argue that, for any global minimizer $(\boldsymbol{w}, b, \boldsymbol{\xi})$, we must have $\xi_i = \max(0, 1 - y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b))\ \forall\ i \in [N]$. Otherwise, there exists a certain $j \in [N]$ so that $\xi_j > \max(0, 1 - y_j(\boldsymbol{w}^\intercal \boldsymbol{x}_j + b))$, but in this case we can replace ξ_j with a smaller ξ_j' so that $\xi_j' = \max(0, 1 - y_j(\boldsymbol{w}^\intercal \boldsymbol{x}_j + b))$ and the objective becomes strictly smaller, contradicting the assumption that $(\boldsymbol{w}, b, \boldsymbol{\xi})$ is a global minimizer. This implies that Eq. (5.1), and in turn Eq. (4.1), is equivalent to the *unconstrained form of soft-margin SVM*:

$$\boxed{\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum\nolimits_{i\in[N]} \max(0, 1 - y_j(\boldsymbol{w}^\intercal \boldsymbol{x}_j + b)).} \tag{5.2}$$

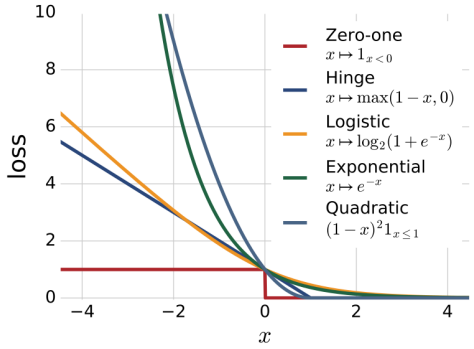


Figure 5: Illustration of the zero-one loss and its various approximations used in machine learning (figure adapted from Fig 7.4 of [Moh18]).

Before we discuss how to solve the unconstrained form, it is worth pondering on the formulation itself. Recall that when we discussed formulating supervised learning problems at the beginning of the chapter on linear predictions, we touched on the generic form of structural risk minimization: for a training set $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i\in[N]}$,

$$\min_{\boldsymbol{\theta}} \frac{1}{N}\sum\nolimits_{i\in[N]} \ell(\boldsymbol{y}_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + \mathcal{R}(\boldsymbol{\theta}), \tag{5.3}$$

where $\boldsymbol{\theta}$ is the parameter of the learning model $f_{\boldsymbol{\theta}}$. The formulation in Eq. (5.2) takes exactly this form: $\frac{1}{2}\|\boldsymbol{w}\|_2^2$ is the regularization on the model parameter $\boldsymbol{w}$, and the terms $\max(0, 1 - y_j(\boldsymbol{w}^\intercal \boldsymbol{x}_j + b))$ for all i's are the individual losses. To see this, for binary classification, we are interested in individual losses $\mathbb{1}\{y_i f_\theta(\boldsymbol{x}_i) < 0\}$, assuming that $y_i \in \{+1, -1\}$ and our actual classifier takes the form $\text{sign}(f_\theta(\boldsymbol{x}_i))$. So, we can view $\max(0, 1 - y_j(\boldsymbol{w}^\intercal \boldsymbol{x}_j + b))$ as the *Hinge loss* (see Fig. 5) with respect to $y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b)$, as an approximation to the zero-one loss on it (i.e., $\mathbb{1}\{y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) < 0\}$). We will encounter other losses later in this course.

5.2 Mini-batch stochastic gradient descent (SGD)

Now, to solve Eq. (5.2), which is unconstrained, we tend to think of the gradient descent method that we learned earlier. There are two catches: (1) the $z \mapsto \max(0, z)$ function is not differentiable at 0; (2) N might be large in modern datasets. Here, issue (1) is not that serious, as we may never

encounter these non-differentiable points during the optimization process. So we will treat it as if it were differentiable[4]. Issue (2) is more intrinsic, and we need new ideas.

Although it is possible to implement (sub)gradient descent to solve Eq. (5.2), we take this opportunity to introduce the powerful (mini-batch) stochastic gradient descent (SGD) method, a workhorse for modern machine learning and deep learning, especially on large-scale datasets.

The central idea is *the law of large numbers*, which says that the sample average $\frac{1}{n}\sum_{i\in[n]} z_i$ of i.i.d. random variables $z_1, \dots, z_n$ converges to its mean as $n \to \infty$. Now consider an optimization problem of the form

$$\min_{\boldsymbol{\theta}} \ \frac{1}{n}\sum\nolimits_{i\in[n]} f(\boldsymbol{\theta}; \boldsymbol{z}_i), \tag{5.4}$$

where $\boldsymbol{\theta}$ is the optimization variable, and $\boldsymbol{z}_i$'s are given data points that are drawn iid from a certain underlying distribution $\mathcal{D}$. Obviously, $\frac{1}{n}\sum_{i\in[n]} f(\boldsymbol{\theta}; \boldsymbol{z}_i) \to \mathbb{E}_{\boldsymbol{z}\sim\mathcal{D}} f(\boldsymbol{\theta}; \boldsymbol{z})$ as $n \to \infty$. To approximate the gradient of $\mathbb{E}_{\boldsymbol{z}\sim\mathcal{D}} f(\boldsymbol{\theta}; \boldsymbol{z})$, we can randomly draw a batch (i.e., set) of points J, where $|J|$ is hopefully way smaller than n, so that

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{z}\sim\mathcal{D}} f(\boldsymbol{\theta}; \boldsymbol{z}) \approx \nabla_{\boldsymbol{\theta}} \frac{1}{|J|}\sum\nolimits_{i\in J} f(\boldsymbol{\theta}; \boldsymbol{z}_i) = \frac{1}{|J|}\sum\nolimits_{i\in J} \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}; \boldsymbol{z}_i), \tag{5.5}$$

where the right side is called a **stochastic gradient**. Then, whenever we need to use the deterministic gradient in a gradient-descent-style algorithm, we replace it by a stochastic gradient. This leads to the mini-batch SGD method, as summarized in Algorithm 1. In practice, randomly sampling

Algorithm 1 Mini-batch stochastic gradient descent (SGD) for solving $\min_{\boldsymbol{\theta}} \ \frac{1}{n}\sum_{i\in[n]} f(\boldsymbol{\theta}; \boldsymbol{z}_i)$

Input: initialization $\boldsymbol{\theta}^{(0)}$, stopping criterion (SC), iteration count $k = 0$

1: **while** SC not satisfied **do**
2: sample a random subset $J_k \subset [n]$
3: calculate a stochastic gradient $\widehat{\boldsymbol{g}_k} \doteq \frac{1}{|J_k|}\sum_{j\in J_k} \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^{(k)}; \boldsymbol{z}_j)$
4: decide a step size $t^{(k)}$
5: make a step: $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - t^{(k)}\widehat{\boldsymbol{g}_k}$
6: update iteration count: $k = k + 1$
7: **end while**

a mini-batch each time might still be expensive. So, people typically shuffle the training set and take consecutive batches instead, as summarized in Algorithm 2—here, each pass over the whole training set is called an **epoch**. Now we come to the step sizes and stopping criterion. In contrast to the case of deterministic gradient descent, where we recommend the back-tracking line search as a reliable adaptive step-size rule, we cannot do the same for mini-batch SGD as evaluating the function value at each iteration would incur $O(n)$ cost again—that we try to avoid. A general rule of thumb is to use diminishing step sizes, e.g.,

$$t^{(k)} = \frac{\alpha}{1+\beta k}, \quad t^{(k)} = \alpha e^{-\beta k}, \quad t^{(k)} \text{ piecewise diminishing constant}, \tag{5.6}$$

where α, β are tunable parameters. The state-of-the-art step size schedules for SGD-based neural network training can be found at

https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate.

[4]For rigorous handling of such issues, we need tools from nonsmooth analysis, which can get very technical [CV20, CP21].

Algorithm 2 Practical mini-batch stochastic gradient descent (SGD) for solving $\min_{\boldsymbol{\theta}} \frac{1}{n}\sum_{i\in[n]} f(\boldsymbol{\theta}; \boldsymbol{z}_i)$

Input: initialization $\boldsymbol{\theta}^{(0)}$, stopping criterion (SC), iteration count $k = 0$, batch size B, epoch count $\ell = 0$

while SC not satisfied **do**
  shuffle the index set $[n]$ and divide it into consecutive batches of size B
  for $i \in \{1, \ldots, \#\text{batches}\}$ **do**
    calculate the stochastic gradient $\widehat{\boldsymbol{g}_k}$ based on the i^{th} batch
    decide a step size $t^{(k)}$
    make a step: $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - t^{(k)}\widehat{\boldsymbol{g}_k}$
    update iteration count: $k = k + 1$
  end for
  update epoch count: $\ell = \ell + 1$
end while

5.3 Mini-batch SGD for solving soft-margin SVMs

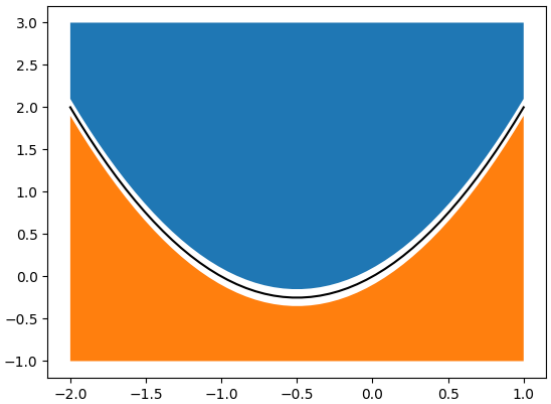
Now, when applying the mini-batch SGD algorithm to solve Eq. (5.2), to calculate a stochastic gradient, one can take the deterministic gradient from the $\frac{1}{2}\|\boldsymbol{w}\|_2^2$ term, plus a stochastic gradient from the $C\sum_{i\in[N]}\max(0, 1 - y_j(\boldsymbol{w}^\intercal \boldsymbol{x}_j + b))$ term via sampling, i.e.,

$$\widehat{\boldsymbol{g}} = \begin{bmatrix}\boldsymbol{w}\\ 0\end{bmatrix} - \frac{NC}{|J|}\sum\nolimits_{j\in J} \mathbb{1}\left\{y_j(\boldsymbol{w}^\intercal \boldsymbol{x}_j + b) \le 1\right\}\begin{bmatrix}y_j\boldsymbol{x}_j\\ y_j\end{bmatrix}, \tag{5.7}$$

where we assume that J is the mini-batch currently sampled.

6 Kernel methods

Although the soft-margin SVM described above allows non-linearly-separable data, it is still restrictive for complex data that might need nonlinear decision boundaries.



blue: positive class orange: negative class
black curve: $x_2 = x_1^2 + x_1$—ideal decision boundary
positive and negative samples not linearly separable

Consider nonlinear mapping of the features:

$$[x_1, x_2] \mapsto [x_1, x_2, x_1^2],$$

the two classes become *linearly separable* in the mapped feature space, as the curve $[1, -1, 1]^\intercal[x_1, x_2, x_1^2] = 0$ perfectly separates them

Figure 6: Example: nonlinear feature mappings can make non-linearly-separable data become linear separable in the mapped feature space.

On the other hand, nonlinear feature mappings, often into higher dimensional spaces, can make non-linearly-separable data become linear separable in the mapped feature spaces, as illustrated in

the quadratic example in Fig. 6: intuitively, by mapping features into collection of their monomials and seeking linear separators in the mapped space, is equivalent to seeking nonlinear decision boundaries defined by polynomial equations in the original space.

We can generalize the idea as follows.

> map all points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N \in \mathbb{R}^d$ into $\Phi(\boldsymbol{x}_1), \dots, \Phi(\boldsymbol{x}_N) \in \mathbb{R}^D$ via a nonlinear mapping $\Phi : \mathbb{R}^d \mapsto \mathbb{R}^D$, and then seek a linear separator in the mapped feature space $\mathbb{R}^D$

For (approximate) linear separation to be possible, often D can be much greater than d, i.e., $D \gg d$. So, storing and computing with the resulting $\Phi(\boldsymbol{x}_1), \dots, \Phi(\boldsymbol{x}_N) \in \mathbb{R}^D$ can be prohibitively expensive. This calls for another idea: **kernel functions**. For any given kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$, it holds that

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle \ \forall \ \boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d \tag{6.1}$$

for a certain Φ. In other words, a kernel function K defines the inner product of every pair of points in a certain mapped feature space, where the mapping Φ is induced by K itself. With this, we do not need to define the nonlinear mapping Φ or compute and store the mapped points, but we can compute the inner product of any mapped point pair.

6.1 Why inner products are sufficient for computation?

But why we only need to worry about inner products in the mapped feature space? Let us first illustrate this using the soft-margin SVM.

Recall from Section 4.2 that any optimal $\boldsymbol{w}$ for the soft-margin SVM can be written as $\boldsymbol{w} = \sum_{i\in[N]} \alpha_i \boldsymbol{x}_i$ for a certain $\boldsymbol{\alpha} \in \mathbb{R}^N$. Substituting this into the unconstrained form of the soft-margin SVM in Eq. (5.2), we obtain that

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^N, b} \ \frac{1}{2}\left\langle \sum\nolimits_{i\in[N]} \alpha_i \boldsymbol{x}_i, \sum\nolimits_{i\in[N]} \alpha_i \boldsymbol{x}_i \right\rangle + C \sum\nolimits_{i\in[N]} \max(0, 1 - y_i(\sum\nolimits_{j\in[N]} \alpha_j \boldsymbol{x}_j^\intercal \boldsymbol{x}_i + b)) \tag{6.2}$$

$$\iff \min_{\boldsymbol{\alpha}\in\mathbb{R}^N, b} \ \frac{1}{2} \sum\nolimits_{i\in[N], j\in[N]} \alpha_i \alpha_j \boldsymbol{x}_i^\intercal \boldsymbol{x}_j + C \sum\nolimits_{i\in[N]} \max(0, 1 - y_i(\sum\nolimits_{j\in[N]} \alpha_j \boldsymbol{x}_j^\intercal \boldsymbol{x}_i + b)) \tag{6.3}$$

$$\iff \min_{\boldsymbol{\alpha}\in\mathbb{R}^N, b} \ \frac{1}{2}\boldsymbol{\alpha}^\intercal \boldsymbol{G} \boldsymbol{\alpha} + C \sum\nolimits_{i\in[N]} \max(0, 1 - y_i(\boldsymbol{g}_i^\intercal \boldsymbol{\alpha} + b)), \tag{6.4}$$

where $\boldsymbol{G} \in \mathbb{R}^{N\times N}$ is defined as $g_{ij} = \boldsymbol{x}_i^\intercal \boldsymbol{x}_j$, and $\boldsymbol{g}_i$ is the i-th column of $\boldsymbol{G}$ by our indexing convention. Note that in the final form of Eq. (6.4), the only way we need to access the data points is through their inner products, i.e., the matrix $\boldsymbol{G}$.

This seems too special; we need KKT analysis to obtain the above for the soft-margin SVM. What if we have nontrivial feature mappings? The following celebrated theorem answers this question.

Theorem 6.1 (Representer theorem)**.** *Consider points $\{\boldsymbol{x}_i\}_{i\in[N]}$ in $\mathbb{R}^d$ and their mapped features $\{\Phi(\boldsymbol{x}_i)\}_{i\in[N]}$ in $\mathbb{R}^D$ via a mapping Φ. Any optimization problem of the form*

$$\min_{\boldsymbol{w}} \ \mathcal{L}(\boldsymbol{w}^\intercal \Phi(\boldsymbol{x}_1), \cdots, \boldsymbol{w}^\intercal \Phi(\boldsymbol{x}_N)) + R(\|\boldsymbol{w}\|_2), \tag{6.5}$$

where $\mathcal{L}$ is arbitrary and $R : \mathbb{R}_+ \mapsto \mathbb{R}$ is monotonically nondecreasing, has an optimal solution[5] that takes the form $\boldsymbol{w}_* = \sum_{i\in[N]} \alpha_i \Phi(\boldsymbol{x}_i)$ for a certain $\boldsymbol{\alpha} \in \mathbb{R}^N$.

[5] There could be other optimal solutions that do not take this form.

Note that here the $\mathbb{R}^D$ here can be generalized to Hilbert spaces, which we avoid to reduce the technicality. Moreover, since we know that an optimal solution takes the form $\boldsymbol{w}_* = \sum_{i\in[N]} \alpha_i \Phi(\boldsymbol{x}_i)$, we can substitute this back to Eq. (6.5), and obtain that

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^N} \mathcal{L}\Big(\sum\nolimits_{i\in[N]} \alpha_i \Phi(\boldsymbol{x}_i)^\intercal \Phi(\boldsymbol{x}_1), \cdots, \sum\nolimits_{i\in[N]} \alpha_i \Phi(\boldsymbol{x}_i)^\intercal \Phi(\boldsymbol{x}_N)\Big) + R\Big(\sqrt{\sum\nolimits_{i\in[N],j\in[N]} \alpha_i \alpha_j \Phi(\boldsymbol{x}_i)^\intercal \Phi(\boldsymbol{x}_j)}\Big), \quad (6.6)$$

where it is again clear that the dependency on the data is only through inner products of the form $\Phi(\boldsymbol{x}_i)^\intercal \Phi(\boldsymbol{x}_j)$.

So, if we can compute the inner product $\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}')\rangle$ for every pair $\boldsymbol{x}, \boldsymbol{x}'$ through a certain function K, i.e., $\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}')\rangle = K(\boldsymbol{x}, \boldsymbol{x}')$. We can solve learning problems of the form Eq. (6.5). This is the essence of **kernel methods**, or **kernel tricks**.

For example, suppose that we select a kernel K and create the **Gram matrix** $\boldsymbol{G} \in \mathbb{R}^{N\times N}$ for training points $\{\boldsymbol{x}_i\}_{i\in[N]}$, we can derive the kernel version of soft-margin SVM as

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^N, b} \frac{1}{2}\boldsymbol{\alpha}^\intercal \boldsymbol{G}\boldsymbol{\alpha} + C\sum\nolimits_{i\in[N]} \max(0, 1 - y_i(\boldsymbol{g}_i^\intercal \boldsymbol{\alpha} + b)), \quad (6.7)$$

which is no different than Eq. (6.4). To make predictions with the model $(\boldsymbol{\alpha}^*, b^*)$ after training, for any test point $\boldsymbol{x}$,

$$\sum\nolimits_{i\in[N]} \alpha_i^* \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x})\rangle + b^* = \sum\nolimits_{i\in[N]} \alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x}) + b^*. \quad (6.8)$$

6.2 Examples of kernel functions

There are a few popularly used kernel functions in practice.

- **linear kernel** $K(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}'\rangle$, the simplest—no feature mapping (or identity feature mapping)
- **polynomial kernel** $K(\boldsymbol{x}, \boldsymbol{x}') = (c + \langle \boldsymbol{x}, \boldsymbol{x}'\rangle)^k$ for a certain $c > 0$, where k is the degree of the polynomial
- **Gaussian (or radial basis function, RBF) kernel** $K(\boldsymbol{x}, \boldsymbol{x}') = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2}{2\sigma^2}}$, where $\sigma > 0$ is the bandwidth parameter

Among them, the Gaussian/RBF kernel is typically considered as the most powerful. The reason is that the effective mapping Φ is actually into an infinite-dimensional space. To see this, we take a simplified version and consider the kernel $K(x, x') = e^{-(x-x')^2/2}$ that operates only on scalar features. We have

$$e^{-(x-x')^2/2} = e^{-x^2/2} e^{-x'^2/2} e^{xx'} = e^{-x^2/2} e^{-x'^2/2} \sum\nolimits_{k=0}^{\infty} \frac{(xx')^k}{k!} = \sum\nolimits_{k=0}^{\infty} \frac{e^{-x^2/2} x^k}{\sqrt{k!}} \frac{e^{-x'^2/2} x'^k}{\sqrt{k!}}, \quad (6.9)$$

which implies that the mapping is $\Phi(x) = e^{-x^2/2}[1,\ x,\ x^2/\sqrt{2!},\ x^3/\sqrt{3!},\ \ldots]$.

One way to think about kernel functions is that they define pairwise similarities. This is clear for the linear kernel, as from linear algebra we know that the inner product of two vectors measures their similarities (perhaps after normalization by their respective lengths). Similarly, for the polynomial and Gaussian kernels, the kernel values become larger when $\boldsymbol{x}$ and $\boldsymbol{x}'$ get closer.

6.3 Symmetric positive definite kernels

K has to satisfy certain conditions to be a valid kernel function, i.e., able to induce a Φ and a valid inner product as described above. One sufficient condition is *symmetric positive definiteness* (SPD): K is said to be *symmetric positive definite* if: 1) it is symmetric, i.e., $K(\boldsymbol{x}, \boldsymbol{x}') = K(\boldsymbol{x}', \boldsymbol{x})$ for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$, and 2)it is positive definite, i.e., for all $m \in \mathbb{N}$ and for all $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$, the Gram matrix $[K(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{ij}$ is *positive semidefinite*[6].

There are several useful operation rules to help tell SPD kernels from elementary compositions of SPD kernels.

- **Summation**: $K_1 + K_2$ is SPD if K_1 and K_2 are SPD.
- **Product**: $K_1 K_2$ is SPD if K_1 and K_2 are SPD. For example, we know $K(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle$ is SPD, so is $K(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle + c$ for any $c \geq 0$, as $\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + c = \langle [\boldsymbol{x}; \sqrt{c}], [\boldsymbol{x}'; \sqrt{c}] \rangle$. So the polynomial kernel $K(\boldsymbol{x}, \boldsymbol{x}') = (c + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^k$ for $k \in \mathbb{N}$ is an SPD kernel by the product rule.
- **Pointwise limit**: the limit $K = \lim_{n \to \infty} K_n$ is SPD if all K_n's are SPD.
- **Power series composition**: $\sum_{n=0}^{\infty} a_n K^n$ is SPD if K is SPD, $a_n \geq 0$ for all n, and K takes values within the convergence radius of the power series $\sum_{n=0}^{\infty} a_n x^n$ (this can easily be shown from the previous three rules).
- **Tensor product/summation**: if K_1 is an SPD kernel on $\mathbb{R}^d$ and K_2 is an SPD kernel on $\mathbb{R}^{d'}$. Then, both $K_1 K_2$ and $K_1 + K_2$ are SPD kernels on $\mathbb{R}^d \times \mathbb{R}^{d'}$. As an application, recall that $K(x, x') = e^{-(x-x')^2/2}$ is an SPD kernel on scalar features. Now consider the Gaussian kernel with $\sigma = 1$ on $\mathbb{R}^d$:

$$\exp\Big(-\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2/2\Big) = \prod_{i=1}^{d} \exp\Big(-(\boldsymbol{x}_i - \boldsymbol{x}_i')^2/2\Big), \tag{6.10}$$

  which is a tensor product of the coordinate-wise SPD kernels, and thus an SPD kernel.

7 Beyond binary classification

Kernel methods can be integrated into numerous learning settings beyond binary classification, which we briefly sample here.

7.1 Multiclass classification

Consider classification problems with K target classes, where $K \geq 3$. A classic idea to solve such problems is to reduce it to a sequence of binary problems, train binary classifiers for these problems *separately*, and then derive a final decision rule based on these classifiers. The upside of such ideas is simplicity, but the downside is the lack of synergies between these binary classifiers, as we explain below. In contrast, modern multiclass classifiers, especially in deep learning, are typically trained jointly.

One-vs-rest & one-vs-one approaches In the one-vs-rest[7] approach, one trains K binary classifiers separately, attempting to separate any particular class vs. the rest classes. In other words, one tries to separate

[6] Yes, this is not a typo but an inconsistency of conventions between different fields: kernels and their positive definiteness are notions commonly used in functional analysis and operator theory, whereas positive (semi)definiteness of matrices in linear algebra and matrix analysis. Warning: different authors use different conventions of positive definiteness in the kernel method literature; make sure you understand their conventions before trying to digest their results.

[7] Or one-vs-all by some authors, which might sound a bit misleading.

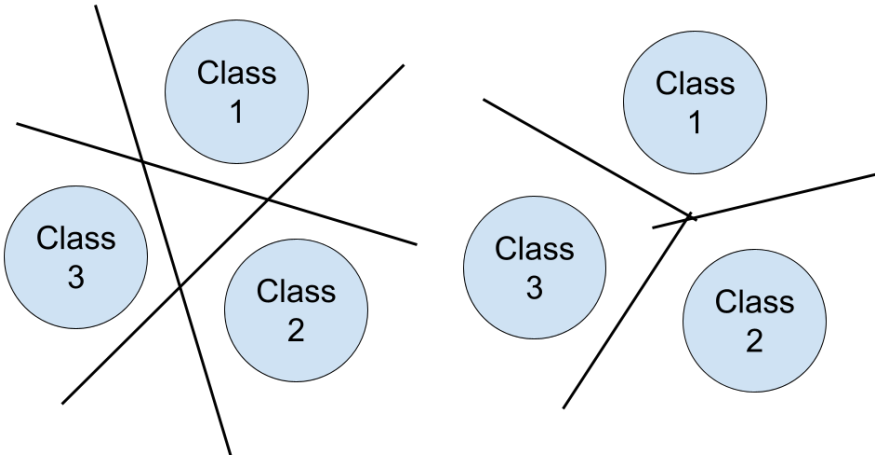


Figure 7: Illustration of the one-vs-rest and the one-vs-one approach on a synthetic 3-class classification problem. The decision boundaries by the associated binary linear classifiers are drawn as solid lines.

class 1 vs. non class 1 class 2 vs. non class 2 ...class K vs. non class K,

as illustrated in Fig. 7 (left). The final decision rule is

$$\arg\max_{\ell \in [K]} f_\ell(\boldsymbol{x}), \tag{7.1}$$

where, for each $\ell \in [K]$, the f_ℓ's is the trained binary classifier which is assumed to output the "confidence" that the current point $\boldsymbol{x}$ belongs to class ℓ.

In contrast, in the one-vs-one approach, one trains $\binom{K}{2}$ binary classifiers, trying to separate every pair of classes, as illustrated in Fig. 7 (right). The final decision rule is typically through majority voting, i.e., for any input $\boldsymbol{x}$, the class receives the most votes by the $\binom{K}{2}$ classifiers is selected (ties broken randomly, or maybe by aggregation of confidence scores).

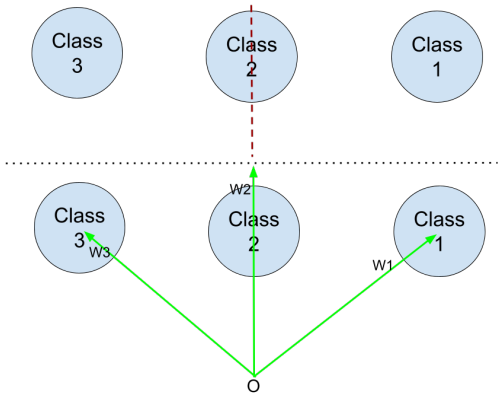


Figure 8: The one-vs-rest approach can be intrinsically suboptimal for multi-class problems.

Although one-vs-rest is simple, there are several issues about it: (1) **incompatible or incomparable confidence scores**, as the classifiers are trained separately. This makes the decision rule as stated in Eq. (7.1) problematic. Although in binary classification we tend to think of the prediction output as approximating $p(y|\boldsymbol{x})$—class confidence, the reality may be far off. This touches on the **calibration issue**, a topic of intensive research; see, e.g., https://scikit-learn.org/1.5/modules/calibration.html, or the **In classification** subsection of https://en.wikipedia.org/wiki/Calibration_(statistics); (2) **imbalanced learning** refers to classification problems with different class frequencies[8]. The one-vs-rest strategy introduces artificially imbalanced learning problems, especially when K is large. Although heuristic methods such as reweighting and resampling are popularly used to handle imbalanced learn-

[8]One can generalize the notion for regression and other learning settings as well.

ing[9], they can be very suboptimal [PTZ+22]; (3) **suboptimality** for certain scenarios. For example, in Fig. 8, we have a 3-class problem where the 3 classes are perfectly separate as 3 spherical clusters. Using the one-vs-rest approach with linear classifiers, we make at least 50% mistakes (achieved by the red separator) when trying to separate class 2 vs classes 1 & 3, even though the other 2 classifiers can be perfect; see Fig. 8 (top). On the other hand, consider the 3 green unit vectors $\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3$ in Fig. 8 (bottom) that pass through the centers of the 3 classes, respectively. Then, the decision rule with 3 linear classifiers

$$\arg\max_{i\in[3]} \boldsymbol{w}_i^\intercal \boldsymbol{x} \tag{7.2}$$

makes perfect prediction for the 3-class problem.

The one-vs-one approach is also not free of issues. The main one is its computational complexity for large K, as one needs to train $O(K^2)$ binary classifiers. Also, if we need to break ties by the aggregation of confidence scores, e.g., predictor outputs, we fall again to the calibration issue alluded to above.

A joint multiclass SVM formulation Here, we describe an idea that performs multiclass learning in a joint formulation, extending the idea of binary SVMs. We start with an ideal decision rule:

$$\arg\max_{k\in[K]} \boldsymbol{w}_k^\intercal \Phi(\boldsymbol{x}) \tag{7.3}$$

where the kernel mapping $\Phi : \mathbb{R}^d \to \mathbb{R}^D$ as always, $\boldsymbol{W} \in \mathbb{R}^{D\times K}$, and for brevity we omit the offset term. Now, given the training set $S = \{(\boldsymbol{x}_i, y_i)\}_{i\in[N]}$, we have

$$y_i = \arg\max_{k\in[K]} \boldsymbol{w}_k^\intercal \Phi(\boldsymbol{x}_i)\ \forall i \in [N] \iff \boldsymbol{w}_{y_i}^\intercal \Phi(\boldsymbol{x}_i) > \max_{k\in[K]\setminus\{y_i\}} \boldsymbol{w}_k^\intercal \Phi(\boldsymbol{x}_i)\ \forall i \in [N] \tag{7.4}$$

So, assuming separability, we can formulate a feasibility problem

$$\text{find } \boldsymbol{W} \in \mathbb{R}^{D\times K} \quad \text{s.t. } \boldsymbol{w}_{y_i}^\intercal \Phi(\boldsymbol{x}_i) - \max_{k\in[K]\setminus\{y_i\}} \boldsymbol{w}_k^\intercal \Phi(\boldsymbol{x}_i) > 0\ \forall i \in [N]. \tag{7.5}$$

Due to the scale ambiguity of the constraints in $\boldsymbol{W}$, we can fix the scale, emulating the process we derived the hard-margin SVM, as:

$$\min_{\boldsymbol{W}\in\mathbb{R}^{D\times K}} \quad \sum\nolimits_{k\in[K]} \|\boldsymbol{w}_k\|_2^2 \quad \text{s.t. } \boldsymbol{w}_{y_i}^\intercal \Phi(\boldsymbol{x}_i) - \max_{k\in[K]\setminus\{y_i\}} \boldsymbol{w}_k^\intercal \Phi(\boldsymbol{x}_i) \geq 1\ \forall i \in [N], \tag{7.6}$$

i.e., multiclass hard-margin SVM. For the corresponding soft-margin version, we again introduce slack variables and obtain

$$\begin{aligned}\min_{\boldsymbol{W}\in\mathbb{R}^{D\times K},\boldsymbol{\xi}\in\mathbb{R}^N} \quad & \sum\nolimits_{k\in[K]} \|\boldsymbol{w}_k\|_2^2 + C\sum\nolimits_{i\in[N]} \xi_i \\ & \text{s.t. } \boldsymbol{w}_{y_i}^\intercal \Phi(\boldsymbol{x}_i) - \max_{k\in[K]\setminus\{y_i\}} \boldsymbol{w}_k^\intercal \Phi(\boldsymbol{x}_i) \geq 1 - \xi_i, \xi_i \geq 0\ \forall i \in [N].\end{aligned} \tag{7.7}$$

Similar to the binary case, we can also turn this one into an equivalent unconstrained form:

$$\min_{\boldsymbol{W}\in\mathbb{R}^{D\times K},\boldsymbol{\xi}\in\mathbb{R}^N} \sum\nolimits_{k\in[K]} \|\boldsymbol{w}_k\|_2^2 + C\sum\nolimits_{i\in[N]} \max\Big(0, 1 - \boldsymbol{w}_{y_i}^\intercal \Phi(\boldsymbol{x}_i) + \max_{k\in[K]\setminus\{y_i\}} \boldsymbol{w}_k^\intercal \Phi(\boldsymbol{x}_i)\Big), \tag{7.8}$$

yielding *an unconstrained form of multiclass soft-margin SVM*. One can extend the representer theorem to this case, and thereby implement the kernel trick, which we omit here.

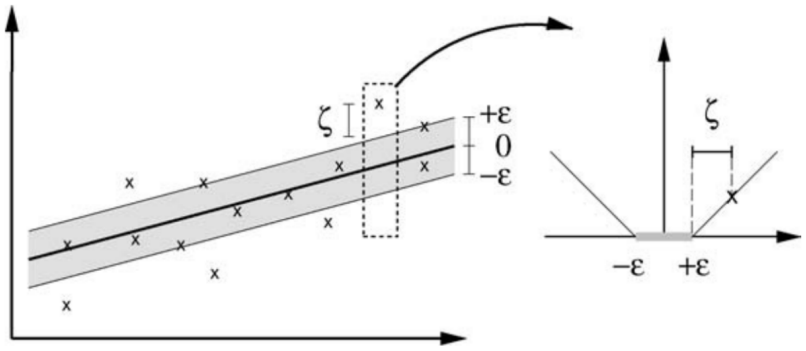


Figure 9: Illustration of support vector regression and the loss it uses

7.2 Support vector regression

For regression, we only consider scalar-valued outputs and assume the training set $S = \{(\boldsymbol{x}_i, y_i)\}_{i\in[N]}$. In classification, we may expect $y_i = f(\boldsymbol{x}_i)$ for all $i \in [N]$, if f is powerful enough. But for regression, this can hardly happen, and we typically only hope that $y_i \approx f(\boldsymbol{x}_i)$ for all $i \in [N]$, or quantitatively $|y_i - f(\boldsymbol{x}_i)| \leq \varepsilon$ for all $i \in [N]$, for a small ε. If we choose a linear prediction model with a kernel mapping Φ, we arrive at

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|_2^2 \quad \text{s.t. } |\boldsymbol{w}^\intercal\Phi(\boldsymbol{x}_i) + b - y_i| \leq \varepsilon \; \forall i \in [N]. \tag{7.9}$$

Now, the constraint is not surprising, but why do we still have the $\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|_2^2$ term in the objective? In other words, why is this a proper extension of the max-margin principle to the regression setting? To resolve this, we need an alternative interpretation of the max-margin principle in the classification setting. Consider our predictor in the mapped feature space, i.e., $f(\Phi(\boldsymbol{x})) = \boldsymbol{w}^\intercal\Phi(\boldsymbol{x}) + b$, and the quantity

$$\begin{aligned}\sup_{\Phi(\boldsymbol{x}),\Phi(\boldsymbol{x}')} \frac{|f(\Phi(\boldsymbol{x})) - f(\Phi(\boldsymbol{x}'))|}{\|\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{x}')\|_2} &= \sup_{\Phi(\boldsymbol{x}),\Phi(\boldsymbol{x}')} \frac{|\boldsymbol{w}^\intercal(\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{x}'))|}{\|\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{x}')\|_2} \\ &\leq \frac{\|\boldsymbol{w}\|_2\|\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{x}')\|_2}{\|\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{x}')\|_2} = \|\boldsymbol{w}\|_2. \end{aligned} \tag{7.10}$$

Here, the upper bound is achievable, e.g., by taking any pair of $(\boldsymbol{x}, \boldsymbol{x}')$ so that $f(\Phi(\boldsymbol{x})) = 1$, $f(\Phi(\boldsymbol{x}')) = -1$, and $\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{x}')$ is parallel to $\boldsymbol{w}$ (in other words, $\Phi(\boldsymbol{x})$ and $\Phi(\boldsymbol{x}')$ are support vectors that are on the positive and negative marginal hyperplanes, respectively, and also their difference is aligned with $\boldsymbol{w}$). Obviously $\|\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{x}')\|_2 = 2/\|\boldsymbol{w}\|_2$—twice the geometric margin, and here maximizing the margin is equivalent to minimizing $\|\boldsymbol{w}\|_2 = \sup_{\Phi(\boldsymbol{x}),\Phi(\boldsymbol{x}')} \frac{|f(\Phi(\boldsymbol{x})) - f(\Phi(\boldsymbol{x}'))|}{\|\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{x}')\|_2}$, which measures the "flatness" of the function $f(\Phi(\boldsymbol{x}))$, or how quickly the output changes with respect to changes to the input—reminding us of gradient: in fact,

$$\|\nabla f(\Phi(\boldsymbol{x}))\|_2 = \|\boldsymbol{w}\|_2. \tag{7.11}$$

This interpretation carries over naturally to the regression setting. One can relax the error tolerance ε by introducing slack variables, yielding the soft-margin version:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum\nolimits_{i\in[N]} \xi_i \quad \text{s.t. } |\boldsymbol{w}^\intercal\boldsymbol{x}_i + b - y_i| \leq \varepsilon + \xi_i, \xi_i \geq 0 \; \forall \in [N], \tag{7.12}$$

[9] see, e.g., https://imbalanced-learn.org/stable/

which is equivalent to the unconstrained formulation

$$\min_{\boldsymbol{w},b} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum\nolimits_{i\in[N]} \max\left(0, |\boldsymbol{w}^\intercal \boldsymbol{x}_i + b - y_i| - \varepsilon\right). \tag{7.13}$$

The soft-margin SVR is illustrated in Fig. 9: geometrically, we try to fit a 2ε-wide tube to the dataset, whose orientation and position are determined by $(\boldsymbol{w}, b)$, while allowing a few outliers to stay outside the tube. Here, these points on and outside the boundaries of the tube correspond to the support vectors.

7.3 One-class SVM

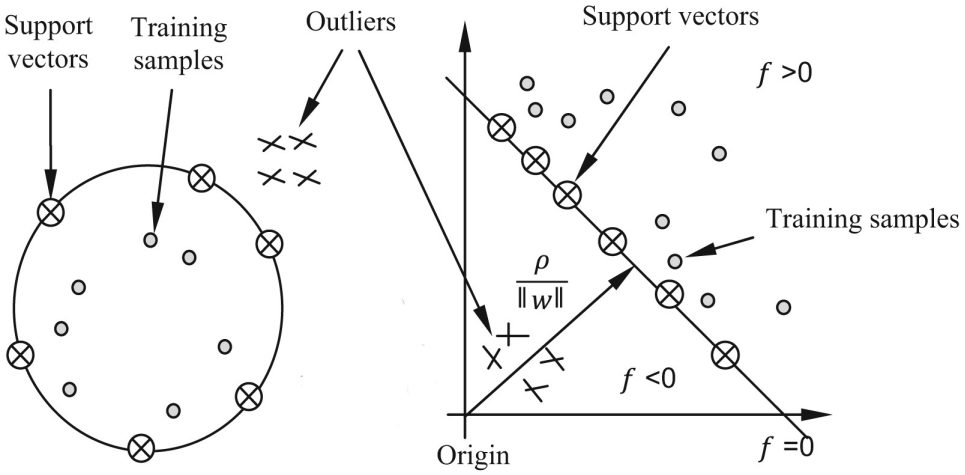


Figure 10: Illustration of one-class SVM with spherical (left) and hyperplane (right) decision boundary, respectively. Figure adapted from Fig 1 of [GCH15].

One-class problems sound very strange, but they are prevalent in practice. For example:

- **novelty detection** We hope to detect future points that are substantially different from all points that we have seen so far, i.e., detection of rare events
- **verification** For example, to verify an individual is the person they claim to be by comparing their facial image against a dataset of their past facial images, i.e., face verification/identification
- **binary classification with an underrepresented class** Intuitively, we need sufficient representation for both classes to perform reasonable binary classification. When one class is deemed poorly represented, modeling the dominant class only might be a better choice.

There are numerous ideas for one-class problems; here, we focus on kernel-based ones. A popular idea: after a kernel mapping, the observed one-class samples are concentrated in a small region. Fig. 10 illustrates a couple of possibilities: in the mapped feature space, the majority of samples are enclosed inside a small sphere (left), or a half-space that is away from the origin.

For the former, a natural formulation is

$$\min_{\boldsymbol{c},R} \ R^2 \quad \text{s.t. } |\Phi(\boldsymbol{x}_i) - \boldsymbol{c}| \le R^2 \ \forall\, i \in [N], \tag{7.14}$$

with a soft-margin version

$$\min_{\boldsymbol{c},R,\boldsymbol{\xi}} \ R^2 + C\sum\nolimits_{i\in[N]} \xi_i \quad \text{s.t. } |\Phi(\boldsymbol{x}_i) - \boldsymbol{c}| \le R^2 + \xi_i, \xi_i \ge 0 \ \forall\, i \in [N]. \tag{7.15}$$

For the latter, a natural formulation is

$$\min_{\boldsymbol{w}} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 \quad \text{s.t. } \boldsymbol{w}^\intercal \Phi(\boldsymbol{x}_i) \geq 1 \ \forall \ i \in [N], \tag{7.16}$$

with a soft-margin version

$$\min_{\boldsymbol{w}} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum\nolimits_{i\in[N]} \xi_i \quad \text{s.t. } \boldsymbol{w}^\intercal \Phi(\boldsymbol{x}_i) \geq 1 - \xi_i, \xi_i \geq 0 \ \forall \ i \in [N], \tag{7.17}$$

or equivalently

$$\min_{\boldsymbol{w}} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum\nolimits_{i\in[N]} \max\left(0, 1 - \boldsymbol{w}^\intercal \Phi(\boldsymbol{x}_i)\right). \tag{7.18}$$

8 Kernel approximation (optional)

Further reading

Chapters 15–16 of [SSS14] and Chapters 5–6 of [Moh18] are our main references. The two monographs [SS02, JST04] are definitive references on SVMs and kernel methods; see also the review paper [HSS08]. [HUL01, SPB04] are excellent textbooks on convex analysis and optimization that are ideal for self-study.

Disclaimer

This set of notes is preliminary and has not been thoroughly proofread. Typos and factual errors are well expected, and hence use it with caution. Bug reports are very welcome and should be sent to Prof. Ju Sun via `jusun@umn.edu`.

A Proof of Lemma 1.1

We will first derive a general result on the ℓ_2 distance between any point $\boldsymbol{x}_0$ and any subspace S.

Let us start with linear subspaces. Let $S = \{\boldsymbol{V}\boldsymbol{\alpha} : \boldsymbol{\alpha} \in \mathbb{R}^k\}$ be k-dimensional linear subspace in $\mathbb{R}^d$, i.e., $\boldsymbol{V} \in \mathbb{R}^{d\times k}$ is a basis for S. Then, for any point $\boldsymbol{x}_0 \in \mathbb{R}^d$, the ℓ_2 distance between $\boldsymbol{x}_0$ and S is defined as

$$d_{\ell_2}(\boldsymbol{x}_0, S) \doteq \min_{\boldsymbol{x}\in S} \ \|\boldsymbol{x}_0 - \boldsymbol{x}\|_2 = \min_{\boldsymbol{\alpha}\in\mathbb{R}^k} \ \|\boldsymbol{x}_0 - \boldsymbol{V}\boldsymbol{\alpha}\|_2. \tag{A.1}$$

Obviously, in terms of finding a global minimizer, we can consider an equivalent problem

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^k} \ f(\boldsymbol{\alpha}) \doteq \|\boldsymbol{x}_0 - \boldsymbol{V}\boldsymbol{\alpha}\|_2^2. \tag{A.2}$$

This is a least-squares problem, and $\boldsymbol{V}$ has full column rank. So there is a unique global minimizer $\boldsymbol{\alpha}_0$. First-order optimality condition yields

$$\boldsymbol{V}^\intercal(\boldsymbol{V}\boldsymbol{\alpha}_0 - \boldsymbol{x}_0) = \boldsymbol{0} \Longrightarrow \boldsymbol{\alpha}_0 = (\boldsymbol{V}^\intercal\boldsymbol{V})^{-1}\boldsymbol{V}^\intercal\boldsymbol{x}_0, \tag{A.3}$$

i.e., $\boldsymbol{x}_{\boldsymbol{V}} \doteq \boldsymbol{V}\boldsymbol{\alpha}_0 = \boldsymbol{V}(\boldsymbol{V}^\intercal\boldsymbol{V})^{-1}\boldsymbol{V}^\intercal\boldsymbol{x}_0 \in S$ is the nearest point to $\boldsymbol{x}_0$ on S as measured by the ℓ_2 distance, and the vector $\boldsymbol{x}_0 - \boldsymbol{x}_{\boldsymbol{V}}$ is orthogonal to S, or equivalently, $\boldsymbol{x}_0 - \boldsymbol{x}_{\boldsymbol{V}}$ is a normal direction for S; see Fig. 11. So

$$d_{\ell_2}(\boldsymbol{x}_0, S) = \|\boldsymbol{x}_0 - \boldsymbol{x}_{\boldsymbol{V}}\|_2 = \left\|\left(\boldsymbol{I} - \boldsymbol{V}(\boldsymbol{V}^\intercal\boldsymbol{V})^{-1}\boldsymbol{V}^\intercal\right)\boldsymbol{x}_0\right\|_2. \tag{A.4}$$

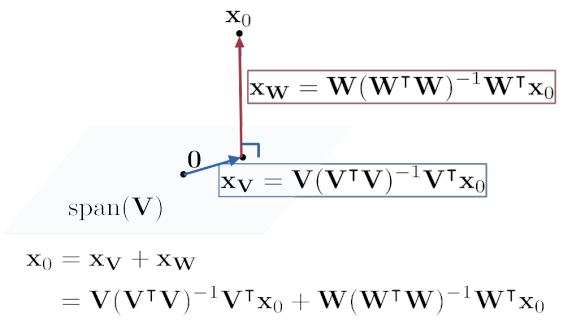


Figure 11: Geometry of orthoprojectors in Euclidean space. $\boldsymbol{V}$ and $\boldsymbol{W}$ together induce an orthogonal decomposition of any given point $\boldsymbol{x}_0$.

Here, $\boldsymbol{x}_{\boldsymbol{V}}$ is the *orthogonal projection* of $\boldsymbol{x}_0$ onto S, and $\mathcal{P}_{\boldsymbol{V}} \doteq \boldsymbol{V}(\boldsymbol{V}^\intercal\boldsymbol{V})^{-1}\boldsymbol{V}^\intercal$ is called the *orthogonal projector*, as $\boldsymbol{x}_{\boldsymbol{V}} = \mathcal{P}_{\boldsymbol{V}}\boldsymbol{x}_0$. Similarly, if $\boldsymbol{W} \in \mathbb{R}^{d\times(d-k)}$ spans the orthogonal subspace $S^\perp$, projection of $\boldsymbol{x}_0$ onto $S^\perp$ is $\boldsymbol{x}_{\boldsymbol{W}} = \mathcal{P}_{\boldsymbol{W}}\boldsymbol{x}_0 = \boldsymbol{W}(\boldsymbol{W}^\intercal\boldsymbol{W})^{-1}\boldsymbol{W}^\intercal\boldsymbol{x}_0$. It is clear we can also write

$$d_{\ell_2}(\boldsymbol{x}_0, S) = \left\|\boldsymbol{W}(\boldsymbol{W}^\intercal\boldsymbol{W})^{-1}\boldsymbol{W}^\intercal\boldsymbol{x}_0\right\|_2. \quad \text{(A.5)}$$

What happens for affine subspaces? Now, suppose that S is an affine subspace and $\boldsymbol{s}_0$ is an arbitrary point on S. For a given point $\boldsymbol{x}_0$, it holds that

$$d_{\ell_2}(\boldsymbol{x}_0, S) = \min_{\boldsymbol{x}\in S}\|\boldsymbol{x}_0 - \boldsymbol{x}\|_2 = \min_{\boldsymbol{x}\in S}\|(\boldsymbol{x}_0 - \boldsymbol{s}_0) - (\boldsymbol{x} - \boldsymbol{s}_0)\|_2 = \min_{\boldsymbol{x}'\in S'}\|(\boldsymbol{x}_0 - \boldsymbol{s}_0) - \boldsymbol{x}'\|_2, \quad \text{(A.6)}$$

where S' is the linear subspace associated with S. Invoking the above result for linear subspaces, we obtain

$$d_{\ell_2}(\boldsymbol{x}_0, S) = \left\|\boldsymbol{W}(\boldsymbol{W}^\intercal\boldsymbol{W})^{-1}\boldsymbol{W}^\intercal(\boldsymbol{x}_0 - \boldsymbol{s}_0)\right\|_2, \quad \text{(A.7)}$$

where $\boldsymbol{W}$ is a basis for the orthogonal subspace of S'.

Now we specialize the result to hyperplanes. Let $L = \{\boldsymbol{x} : \langle \boldsymbol{w}, \boldsymbol{x}\rangle + b = 0\}$, where $b = -\langle \boldsymbol{w}, \boldsymbol{s}_0\rangle$ for an arbitrary point $\boldsymbol{s}_0 \in S$. For any $\boldsymbol{x}_0$, invoking the result of Eq. (A.7), we obtain

$$d_{\ell_2}(\boldsymbol{x}_0, S) = \left\|\boldsymbol{w}\|\boldsymbol{w}\|_2^{-2}\boldsymbol{w}^\intercal(\boldsymbol{x}_0 - \boldsymbol{s}_0)\right\|_2 = \frac{|\boldsymbol{w}^\intercal\boldsymbol{x}_0 - \boldsymbol{w}^\intercal\boldsymbol{s}_0|}{\|\boldsymbol{w}\|_2} = \frac{|\boldsymbol{w}^\intercal\boldsymbol{x}_0 + b|}{\|\boldsymbol{w}\|_2}, \quad \text{(A.8)}$$

completing the proof.

References

[CL11] Chih-Chung Chang and Chih-Jen Lin, *LIBSVM: A library for support vector machines*, ACM Transactions on Intelligent Systems and Technology **2** (2011), 27:1–27:27, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[CP21] Ying Cui and Jong-Shi Pang, *Modern nonconvex nondifferentiable optimization*, SIAM, 2021.

[CV20] Christian Clason and Tuomo Valkonen, *Introduction to nonsmooth analysis and optimization*, arXiv preprint arXiv:2001.00216 (2020).

[GCH15] Yasmine Guerbai, Youcef Chibani, and Bilal Hadjadji, *The effective use of the one-class svm classifier for handwritten signature verification based on writer-independent parameters*, Pattern Recognition **48** (2015), no. 1, 103–113.

[HSS08] Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola, *Kernel methods in machine learning*, The Annals of Statistics **36** (2008), no. 3, 1171–1220.

[HUL01] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal, *Fundamentals of convex analysis*, Springer Berlin Heidelberg, 2001.

[JST04] Nello Cristianini John Shawe-Taylor, *Kernel methods for pattern analysis*, Cambridge University Press, 2004.

[Moh18] Mehryar Mohri, *Foundations of machine learning*, 2 ed., The MIT Press, Cambridge, Massachusetts, 2018.

[PTZ+22] Le Peng, Yash Travadi, Rui Zhang, Ying Cui, and Ju Sun, *Imbalanced classification in medical imaging via regrouping*, arXiv preprint arXiv:2210.12234 (2022).

[SPB04] Lieven Vandenberghe Stephen P. Boyd, *Convex optimization*, CAMBRIDGE, 2004.

[SS02] Bernhard Schölkopf and Alexander J. Smola, *Learning with kernels : support vector machines, regularization, optimization, and beyond*, MIT Press, Cambridge, Mass, 2002.

[SSS14] Shai Ben-David Shai Shalev-Shwartz, *Understanding machine learning*, Cambridge University Press, 2014.