Support Vector Machines

Hanno Reuvers

September 19, 2025

Contents

1 Classifying Linearly Separated Data

`=== Notebook1_Classifying_Linearly_Separated_Data.ipynb ===`

To develop an understanding of Support Vector Machines (SVMs), we will first discuss the stylized problem of classifying linearly separable data.[1] That is, the data contains two classes and there exists a hyperplane in the feature space such that all data points of the same class are located on the same side of the hyperplane. The illustration in Figure 1(a) shows two separating hyperplanes as black dashed/dotted lines. Clearly, separating hyperplanes are non-unique. To identify a single hyperplane, we will focus on the separating hyperplane that maximizes the distance to the nearest data point. Figure 1(b) shows this hyperplane. In the remainder of this section we study the optimization problem that computes the latter hyperplane.

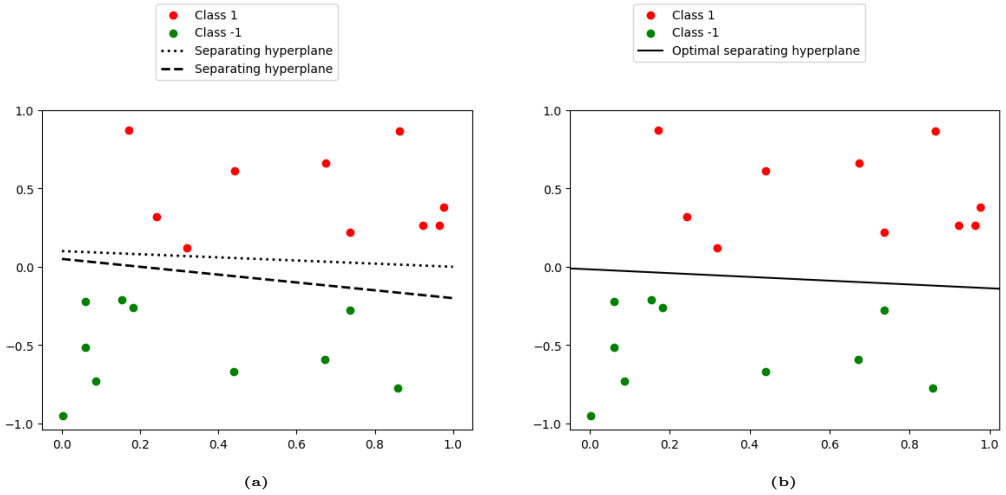

Figure 1: A classification problem with 20 data points equally divided into the class with label 1 (red points) and label -1 (green points). Note that separating hyperplanes in a two-dimensional space are line. **(a)** A plot of the data with two separating hyperplanes indicated by black dashed/dotted lines. **(b)** The black solid line is the "optimal" hyperplane maximizing the distance between the hyperplane and the data point closest to it.

We start with a characterization of a hyperplane. The hyperplane is uniquely determined if we specify a point $\boldsymbol{x}_0$ on the hyperplane and a vector $\boldsymbol{w}$ that is orthogonal/normal to the hyperplane. As illustrated in Figure 2, the vector $\boldsymbol{x}$ is located on the hyperplane if and only if $\boldsymbol{x}-\boldsymbol{x}_0$ is perpendicular to $\boldsymbol{w}$. The latter occurs if $\boldsymbol{w} \cdot (\boldsymbol{x} - \boldsymbol{x}_0) = 0$.[2] Definition 1 below formalizes this notion.

Definition 1 (Point-normal Equation for a Hyperplane)
For some $K > 1$, let $\boldsymbol{w}$, $\boldsymbol{x}_0$, and $\boldsymbol{x}$ denote vectors in $\mathbb{R}^K$. The point-normal equation for a plane with position vector $\boldsymbol{x}_0$ and normal vector $\boldsymbol{w}$ is

$$\boldsymbol{w} \cdot (\boldsymbol{x} - \boldsymbol{x}_0) = 0. \tag{1}$$

[1] These discussions on SVMs are loosely based on the materials in Mohri et al. (2012).

[2] For any vectors $\boldsymbol{a}, \boldsymbol{b}$ of the same size, we let $\boldsymbol{a} \cdot \boldsymbol{b}$ denote the Euclidean inner product between these vectors. That is, if these vectors have length say n, that is $\boldsymbol{a} = (a_1, \ldots, a_n)^\top$ and $\boldsymbol{b} = (b_1, \ldots, b_n)^\top$, then $\boldsymbol{a} \cdot \boldsymbol{b} = \sum_{i=1}^n a_i b_i$. Note that $\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{a}^\top \boldsymbol{b}$.

Definition 2 (Geometric Margin)
The geometric margin at a point $\boldsymbol{\xi} \in \mathbb{R}^K$ *is its Euclidean distance to the hyperplane defined by* $\boldsymbol{w} \cdot (\boldsymbol{x} - \boldsymbol{x}_0) = 0$*:*

$$\rho(\boldsymbol{\xi}) = \frac{|\boldsymbol{w} \cdot (\boldsymbol{\xi} - \boldsymbol{x}_0)|}{\|\boldsymbol{w}\|}, \tag{2}$$

where $\|\boldsymbol{w}\| = \sqrt{\boldsymbol{w} \cdot \boldsymbol{w}} = \sqrt{\sum_{k=1}^{K} w_k^2}$.

The expression for the distance between a point $\boldsymbol{\xi}$ and the hyperplane is given in Definition 2.[3] To develop the intuition behind this result, we refer to the right image in Figure 2. We are looking at the hyperplane sideways and thus see the hyperplane as a solid blue line. The distance between the hyperplane and $\boldsymbol{\xi}$ is given by the length of the orthogonal projection of $\boldsymbol{\xi} - \boldsymbol{x}_0$ onto $\boldsymbol{w}$. This vector component of $\boldsymbol{\xi} - \boldsymbol{x}_0$ along $\boldsymbol{w}$ has length $\|\boldsymbol{\xi} - \boldsymbol{x}_0\| \, |\cos(\theta)|$. With the Euclidean inner product between the vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ being defined as

$$\boldsymbol{u} \cdot \boldsymbol{v} = \begin{cases} \|\boldsymbol{u}\| \, \|\boldsymbol{v}\| \cos(\theta) & \text{if } \boldsymbol{u} \neq \boldsymbol{0} \text{ and } \boldsymbol{v} \neq \boldsymbol{0}, \\ 0 & \text{if } \boldsymbol{u} = \boldsymbol{0} \text{ or } \boldsymbol{v} = \boldsymbol{0}, \end{cases} \tag{3}$$

it follows that $\boldsymbol{w} \cdot (\boldsymbol{\xi} - \boldsymbol{x}_0) = \|\boldsymbol{w}\| \, \|\boldsymbol{\xi} - \boldsymbol{x}_0\| \cos(\theta)$ and thus $\rho(\boldsymbol{\xi}) = \|\boldsymbol{\xi} - \boldsymbol{x}_0\| \, |\cos(\theta)| = \frac{|\boldsymbol{w} \cdot (\boldsymbol{\xi} - \boldsymbol{x}_0)|}{\|\boldsymbol{w}\|}$.

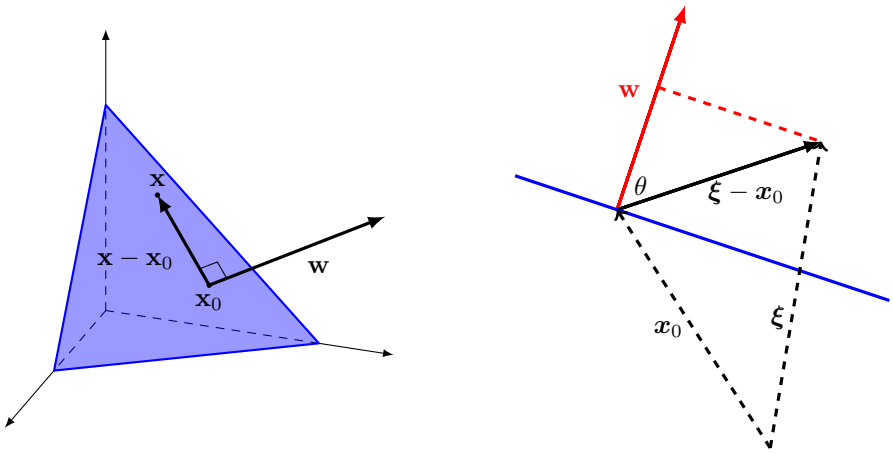

Figure 2: Visualizations to illustrate the concepts of the point-normal equation for a hyperplane (left) and geometric margin (right).

We consider the dataset $\mathcal{S} = \big((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\big)$. That is, for each $i \in \{1, \ldots, n\}$, we observe the feature vector $\boldsymbol{x}_i \in \mathbb{R}^K$ and the class label $y_i \in \{-1, 1\}$. The optimization problem that we are about to define has two requirements:

1. The constraints should ensure that the constructed hyperplane is a separating hyperplane.
2. The distance to the point closest to the hyperplane should be maximized.

[3] Chapter 3 in Anton and Rorres (2013) contains an excellent exposition of the various linear algebra concepts that are useful when studying geometry.

The plots in Figure 3 guide the implementation of the first requirement. The figure on the left shows the angle θ between the normal vector to the plane $\boldsymbol{w}$ and the vector $\boldsymbol{x} - \boldsymbol{x}_0$. The right figure plots $\cos(\theta)$. If we move around the point $\boldsymbol{x}$ in the red area, then $\theta \in (-\pi/2, \pi/2)$ and $\cos(\theta) > 0$. Conversely, placing $\boldsymbol{x}$ in the green area will result in $\cos(\theta) < 0$. We conclude that $\cos(\theta)$ and thus by (3) also $\boldsymbol{w}^\top(\boldsymbol{x} - \boldsymbol{x}_0)$ is positive (negative) on the side of the hyperplane indicated by (opposite to) the normal vector $\boldsymbol{w}$. This property is combined with the class labels being in $\{-1, 1\}$. That is, requiring

$$y_i \boldsymbol{w} \cdot (\boldsymbol{x}_i - \boldsymbol{x}_0) > 0 \qquad \text{for } i = 1, \ldots, n, \tag{4}$$

we ensure that the hyperplane is separating the two classes.

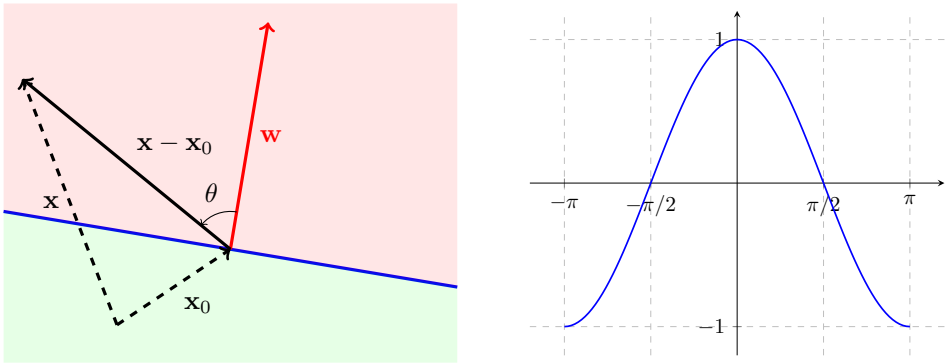

Figure 3: The sign of the inner product $\boldsymbol{w} \cdot (\boldsymbol{x} - \boldsymbol{x}_0) = \|\boldsymbol{w}\| \, \|\boldsymbol{x} - \boldsymbol{x}_0\| \cos(\theta)$ is determined by the sign of $\cos(\theta)$. The right figure shows that $\cos(\theta)$ is non-negative for $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$. The side of the hyperplane to which $\boldsymbol{w}$ is pointing contains points $\boldsymbol{x}$ for which $\boldsymbol{w} \cdot (\boldsymbol{x} - \boldsymbol{x}_0)$ is positive. This region is indicated in red in the left figure. All points in the green area are characterized by $\boldsymbol{w} \cdot (\boldsymbol{x} - \boldsymbol{x}_0) < 0$.

The objective function follows from Definition 2. For any given hyperplane specified by $\boldsymbol{w}$ and $\boldsymbol{x}_0$, its distance to the data point $\boldsymbol{x}_i$ is $\rho(\boldsymbol{x}_i) = \frac{|\boldsymbol{w} \cdot (\boldsymbol{x}_i - \boldsymbol{x}_0)|}{\|\boldsymbol{w}\|}$. The closest point to the hyperplane is thus $\min_{i=1,\ldots,n} \frac{|\boldsymbol{w} \cdot (\boldsymbol{x}_i - \boldsymbol{x}_0)|}{\|\boldsymbol{w}\|}$ and this quantity is to be maximized. Collecting objective function and constraints, the optimization problem is as follows:

$$\begin{aligned} \max_{\boldsymbol{w} \in \mathbb{R}^K, \boldsymbol{x}_0 \in \mathbb{R}^K} \quad & \min_{i=1,\ldots,n} \frac{|\boldsymbol{w} \cdot (\boldsymbol{x}_i - \boldsymbol{x}_0)|}{\|\boldsymbol{w}\|}, \\ \text{subject to:} \quad & y_i \boldsymbol{w} \cdot (\boldsymbol{x}_i - \boldsymbol{x}_0) \geq 0 \text{ for } i = 1, \ldots, n. \end{aligned} \tag{5}$$

The optimization problem in (5) is over-parameterized. That is, the problem depends solely on expressions like $\boldsymbol{w} \cdot (\boldsymbol{x}_i - \boldsymbol{x}_0) = \boldsymbol{w} \cdot \boldsymbol{x}_i - \boldsymbol{w} \cdot \boldsymbol{x}_0$ and many combinations of $(\boldsymbol{w}, \boldsymbol{x}_0) \in \mathbb{R}^K \times \mathbb{R}^K$ yield the same solution.[4] The solution is straightforward. We set $b = -\boldsymbol{w} \cdot \boldsymbol{x}_0$ and rewrite (5) as

$$\begin{aligned} \max_{\boldsymbol{w} \in \mathbb{R}^K, b \in \mathbb{R}} \quad & \min_{i=1,\ldots,n} \frac{|\boldsymbol{w} \cdot \boldsymbol{x}_i + b|}{\|\boldsymbol{w}\|}, \\ \text{subject to:} \quad & y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 0 \text{ for } i = 1, \ldots, n. \end{aligned} \tag{6}$$

[4]Consider $K = 2$ and stack the optimization variables as $\boldsymbol{\zeta} = (w_1, w_2, x_{0,1}, x_{0,2})^\top$. If $w_2 \neq 0$, then, as an example, the sets of optimization variables $\boldsymbol{\zeta}$ and $\boldsymbol{\zeta}^* = (w_1, w_2, x_{0,1} + 1, x_{0,2} - \frac{w_1}{w_2})$. Even when the elements in $\boldsymbol{w}$ are kept fixed, there are simply many choices for $\boldsymbol{x}_0$ that yield the same value for the linear combination $\boldsymbol{w}^\top \boldsymbol{x}_0$.

1.1 Rewriting the Optimization Problem

The optimization problem in (6) can be simplified considerably. First, we exploit the constraints and $y_i \in \{-1, 1\}$ to rewrite the absolute value. Also, since $\boldsymbol{w}$ does not depend on i, we have

$$\begin{aligned} \max_{\boldsymbol{w},b} \quad & \frac{1}{\|\boldsymbol{w}\|} \min_{i=1,\dots,n} y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b), \\ \text{subject to:} \quad & y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 0 \text{ for } i = 1, \dots, n. \end{aligned} \tag{7}$$

We subsequently focus on the expression $\frac{1}{\|\boldsymbol{w}\|} \min_{i=1,\dots,n} y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b)$. For any constant $C > 0$, this expression does not change when replacing $(b, \boldsymbol{w})$ by $(Cb, C\boldsymbol{w})$. We are thus free to choose a scale ourselves. A convenient choice is the rescaling that imposes $\min_{i=1,\dots,n} y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) = 1$. The maximization problem now becomes

$$\begin{aligned} \max_{\boldsymbol{w},b} \quad & \frac{1}{\|\boldsymbol{w}\|}, \\ \text{subject to:} \quad & y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 \text{ for } i = 1, \dots, n. \end{aligned} \tag{8}$$

As a final adjustment, we minimize $\frac{1}{2}\|\boldsymbol{w}\|^2$ instead of maximizing $\frac{1}{\|\boldsymbol{w}\|}$. Without changing the solution, this simplifies the analysis of the optimization problem.

SVM Optimization Problem (Separable case)

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2}\|\boldsymbol{w}\|^2, \tag{9a}$$

$$\text{subject to:} \quad y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 \qquad \text{for } i = 1, \dots, n \tag{9b}$$

Remark 1 (Uniqueness of the Solution)
The optimization problem defined by (9a)–(9b) is quadratic (and hence convex) with linear constraints in the optimization parameters. The feasible set, which can be the empty set, is thus a convex set. These convex optimization problems have the desirable property that a local optimum must also be a global optimum, see e.g. section 4.2.2 in Boyd and Vandenberghe (2004). That is, if a solution to the SVM Optimization Problem exists, then this solution is necessarily unique. Any local optimum must be the global optimum.

Remark 2 (Quadratic Program to Determine the Separating Hyperplane)
Designated solvers for convex quadratic programs (QPs) are readily available, e.g. the `qpsolvers` *library in Python. The latter library solves a QP of the standard form*

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^\top \boldsymbol{x}, \\ \textit{subject to:} \quad & \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{h}, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \\ & lb \leq \boldsymbol{x} \leq ub, \end{aligned} \tag{10}$$

where all inequalities should be interpreted entry-wise. It should be stressed that the vector $\boldsymbol{x}$ in (10) contains all optimization variables. Specifically, we have $\boldsymbol{x} = (\boldsymbol{w}^\top, b)^\top$. Regarding the objective function, it suffices to set $\boldsymbol{P} = \begin{pmatrix} \boldsymbol{I}_K & \boldsymbol{0} \\ \boldsymbol{0}^\top & 0 \end{pmatrix}$ and $\boldsymbol{q} = \boldsymbol{0}$. Also, if (9b) is multiplied by -1, then we have $-y_i \begin{bmatrix} \boldsymbol{x}_i^\top & -1 \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ b \end{bmatrix} \leq -1$. The i^{th} row of $\boldsymbol{G}$ should thus be set equal to $-y_i \begin{bmatrix} \boldsymbol{x}_i^\top & -1 \end{bmatrix}$ and $\boldsymbol{h} = -\boldsymbol{1}$. There is no need to pass the quantities $\boldsymbol{A}$, $\boldsymbol{b}$, lb and ub to `qpsolvers` *as their default values automatically omit the corresponding constraints.*

2 Support Vector Machines (SVMs)

`=== Notebook2_Support_Vector_Machines.ipynb ===`

If the data cannot be separated linearly, then the approach of the previous section will not work. Specifically, there is no possibility to satisfy the set of constraints in (9b) and no feasible solution can be found. To have at least a feasible solution, we need to relax the constraints. As such, we define the slack variables $\xi_i \geq 0$ $(i = 1. \ldots, n)$ and alter the constraints to

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i \qquad \text{for } i = 1, \ldots, n. \tag{11}$$

These new constraints are indeed useful when the data cannot be separated linearly. To see this, we refer to the discussion leading to (4) and the left visualization in Figure 3. If all observations with label $+1$ can be located in the red area (i.e. the direction of the hyperplane in which the normal vector $\boldsymbol{w}$ is pointing) and all observations with label -1 can be located in the green area, then $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) > 0$ for all $i = 1, \ldots, n$. If an observation would be located on the wrong side of the hyperplane ($y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) < 0$), then we can now allow it by having its corresponding slack variable take a value larger than 1.

To avoid many misclassified observations, we should try to find solutions for which the slack variables $\xi_1, \ldots, \xi_n$ do not take large values. To accomplish this, the optimization problem that defines SVMs is the following.

SVM Optimization Problem (Primal)

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} \xi_i, \tag{12a}$$

$$\text{subject to:} \quad y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i \qquad \text{for } i = 1, \ldots, n \tag{12b}$$

$$\xi_i \geq 0 \tag{12c}$$

This optimization problem contains the two hyperparameters C and p. We make the following remarks:

- Since $\xi_i \geq 0$, we can also write $\sum_{i=1}^{n} \xi_i = \sum_{i=1}^{n} |\xi_i| = \|\boldsymbol{\xi}\|_1$, where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)^\top$ and $\|\cdot\|_1$ denotes the L_1-norm.[5] Actually, to preserve the convexity of the optimization problem (see Remark 1 for its importance), we could consider an objective functions $\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\|\boldsymbol{\xi}\|_p$ for some $p \geq 1$. However, $p = 1$ is the most widely used case.
- The hyperparameter C balances the maximization of the margin (through the minimization of $\frac{1}{2}\|\boldsymbol{w}\|^2$) against the norm of the slack parameter vector. A low (high) of C maximizes the margin while allowing for more (less) misclassifications. The value of C is typically determined by a grid search involving cross-validation or the performance on the validation data set.

Remark 3 (Hinge Loss)
The constraints of the SVM Optimization Problem can be incorporated into the objective function. For each $i \in \{1, \ldots, n\}$, we have

$$\xi_i \geq 1 - y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \qquad \textit{and} \qquad \xi_i \geq 0. \tag{13}$$

[5] For a vector $\boldsymbol{x} = (x_1, \ldots, x_n)^\top \in \mathbb{R}^n$, its L_p-norm is defined as $\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$. If we set $p = 1$, then $\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i|$.

As ξ_i enters directly into the objective function, this contribution is either $1 - y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b)$ in case this quantity is non-negative, or zero. Exploiting this property, the optimization problem can be rewritten as

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} \max\left\{0, 1 - y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b)\right\}. \tag{14}$$

Assuming $C > 0$, we set $C = \frac{1}{\lambda n}$ and multiply (14) with $\lambda > 0$. The result is

$$\min_{\boldsymbol{w},b} \frac{1}{n}\sum_{i=1}^{n} \max\left\{0, 1 - y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b)\right\} + \frac{1}{2}\lambda\|\boldsymbol{w}\|^2. \tag{15}$$

From a more classical perspective, the objective function in (15) is the sum of a measure of fit and a L_2 regularization term with penalty parameter λ. If we define hinge loss function $h(x) = \max\{0, 1 - x\}$, then the measure of fit is the sample average $\frac{1}{n}\sum_{i=1}^{n} h\big(y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b)\big)$. Section 12.3.2 in Hastie et al. (2009) contains a discussion of the hinge loss function in comparison to other two-class classification loss functions.

Remark 4 (Quadratic Program for the Primal Solution of the SVM)
Define the vector of solution variables as $\boldsymbol{x} = (\boldsymbol{w}^\top, b, \boldsymbol{\xi}^\top)^\top$. The length of this vector is $(K+1+n)$. Rearranging terms, the i^{th} constraint becomes $-y_i(\boldsymbol{x}_i \cdot \boldsymbol{w}) - y_i b - \xi_i \leq -1$. The full set of constraints can be expressed as

$$\begin{bmatrix} -y_1\boldsymbol{x}_1^\top & -y_1 & -1 & 0 & \cdots & 0 \\ -y_2\boldsymbol{x}_2^\top & -y_2 & 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -y_n\boldsymbol{x}_n^\top & -y_n & 0 & 0 & \cdots & -1 \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ b \\ \boldsymbol{\xi} \end{bmatrix} \leq \begin{bmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{bmatrix}. \tag{16}$$

Defining $\boldsymbol{y} = (y_1, \ldots, y_n)^\top \in \mathbb{R}^n$ and $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^\top \\ \vdots \\ \boldsymbol{x}_n^\top \end{bmatrix} \in \mathbb{R}^{n \times K}$, we can express (16) more compactly as

$$\begin{bmatrix} -\operatorname{diag}(\boldsymbol{y})\boldsymbol{X} & -\boldsymbol{y} & -\boldsymbol{I}_n \end{bmatrix} \boldsymbol{x} \leq -\boldsymbol{\imath}.^{6} \tag{17}$$

The constraints $\xi_i \geq 0$ $(i = 1, \ldots, n)$ can be implemented by setting the appropriate elements of lb equal to zero. To implement the objective function, it suffices to set

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{I}_K & \boldsymbol{O}_{K\times(n+1)} \\ \boldsymbol{O}_{(n+1)\times K} & \boldsymbol{O}_{(n+1)\times(n+1)} \end{bmatrix}, \qquad \text{and} \qquad \boldsymbol{q} = \begin{bmatrix} \boldsymbol{0}_{K+1} \\ \boldsymbol{\imath}_n \end{bmatrix}. \tag{18}$$

2.1 The Lagrangian and the Solution's Interpretation

We introduce the vectors $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^\top \in \mathbb{R}^n_+$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)^\top \in \mathbb{R}^n_+$ containing Lagrange multipliers. The Lagrangian for the *SVM Optimization Problem* becomes

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) - 1 + \xi_i\right] - \sum_{i=1}^{n}\beta_i\xi_i. \tag{19}$$

[6] For a vector $\boldsymbol{x} = (x_1, \ldots, x_n)^\top \in \mathbb{R}^n$, we denote by $\operatorname{diag}(\boldsymbol{x})$ the $(n\times)n$ diagonal matrix which has the elements of $\boldsymbol{x}$ on its main diagonal. Instead of calculating $\operatorname{diag}(\boldsymbol{y})\boldsymbol{X}$ explicitly, it is often more efficient to rely on broadcasting. For example, if $\boldsymbol{y}$ and $\boldsymbol{X}$ have the shapes indicated above, then the NumPy expression `y*X` will directly calculate $\operatorname{diag}(\boldsymbol{y})\boldsymbol{X}$.

At the solution, the derivatives with respect to all the primal variables should be equal to zero. We should have $\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \mathbf{0}$, $\frac{\partial \mathcal{L}}{\partial b} = 0$, and $\frac{\partial \mathcal{L}}{\partial \xi_i} = \mathbf{0}$ for all $i = 1, \ldots, n$. When calculating these derivatives, the comments in the following remark can be useful.

Remark 5 (Calculating Gradients of Linear and Quadratic Forms)
Let $\boldsymbol{x} = (x_1, \ldots, x_n)^\top \in \mathbb{R}^n$ and consider the function $f : \mathbb{R}^N \to \mathbb{R}$. The gradient $\frac{\partial f}{\partial \boldsymbol{x}}$ is a n-dimensional vector with its i^{th} element being equal to $\left[\frac{\partial f}{\partial \boldsymbol{x}}\right]_i = \frac{\partial f}{\partial x_i}$ for all $i = 1, \ldots, n$. Two illustrative examples are provided.
First, we consider the linear form $\boldsymbol{x}^\top \boldsymbol{a}$ for some fixed vector $\boldsymbol{a} = (a_1, \ldots, a_n)^\top$. We have $\frac{\partial}{\partial x_i} \boldsymbol{x}^\top \boldsymbol{a} = \frac{\partial}{\partial x_i} \sum_{j=1}^n x_j a_j = a_i$, and conclude that $\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{a} = \boldsymbol{a}$. For the second example, we take a fixed matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ and construct the quadratic form

$$\boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x} = \sum_{j=1}^n \sum_{\ell=1}^n x_j A_{j\ell} x_\ell = \sum_{j=1}^n A_{jj} x_j^2 + \sum_{j=1}^n \sum_{\ell=1, \ell \neq j}^n x_j A_{j\ell} x_\ell$$

and find

$$\begin{aligned} \frac{\partial}{\partial x_i} \boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x} &= 2A_{ii} x_i + \sum_{\ell=1, \ell \neq i}^n A_{i\ell} x_\ell + \sum_{j=1, j \neq i}^n x_j A_{ji} = \sum_{j=1}^n A_{ij} x_j + \sum_{j=1}^n [\boldsymbol{A}^\top]_{ij} x_j \\ &= [\boldsymbol{A}\boldsymbol{x}]_i + \left[\boldsymbol{A}^\top \boldsymbol{x}\right]_i . \end{aligned}$$

The result is $\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{A}^\top \boldsymbol{x}$. If $\boldsymbol{A}$ is symmetric ($\boldsymbol{A} = \boldsymbol{A}^\top$), then $\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x} = 2\boldsymbol{A}\boldsymbol{x}$.

Setting the derivatives to zero, we find:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^n \alpha_i y_i \boldsymbol{x}_i \qquad \Longrightarrow \qquad \boldsymbol{w} = \sum_{i=1}^n \alpha_i y_i \boldsymbol{x}_i, \tag{20}$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^n \alpha_i y_i = 0 \qquad \Longrightarrow \qquad \sum_{i=1}^n \alpha_i y_i = 0, \tag{21}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \qquad \Longrightarrow \qquad \alpha_i + \beta_i = C. \tag{22}$$

For $i = 1, \ldots, n$, the following complementary conditions should hold as well:

$$\alpha_i \left[y_i (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) - 1 + \xi_i \right] = 0 \qquad \Longrightarrow \qquad \alpha_i = 0 \text{ or } y_i (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) = 1 - \xi_i, \tag{23}$$

$$\beta_i \xi_i = 0 \qquad \Longrightarrow \qquad \beta_i = 0 \text{ or } \xi_i = 0. \tag{24}$$

According to (20), the solution for $\boldsymbol{w}$ is a linear combination of the feature vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$. If $\alpha_i = 0$, then the corresponding $\boldsymbol{x}_i$ does not contribute to the linear combination. The collection of vectors for which the corresponding alphas are unequal to zero are called *support vectors*. Because $\alpha_i \neq 0$ for these support vectors, we must have $y_i (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) = 1 - \xi_i$ by (23). The support vectors can now be grouped into two categories:

1. The support vectors for which $\xi_i = 0$, that is, the support vectors that are on the marginal hyperplanes.

2. Vectors with $\xi_i > 0$. These vectors are either misclassified ($\xi_i > 1$) or located within the margin ($0 < \xi_i \leq 1$). From $\beta_i \xi_i = 0$, it follows that these outliers must have $\alpha_i = C$.

2.2 Deriving the Dual Problem

The dual of the SVM Optimization Problem is found by substituting the solution for $\boldsymbol{w}$, see (20), back into the Lagrangian:

$$\begin{aligned}\mathcal{L} &= \frac{1}{2}\left\|\sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i\right\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i y_i\left(\sum_{j=1}^{n}\alpha_j y_j \boldsymbol{x}_j\right)\cdot \boldsymbol{x}_i + \sum_{i=1}^{n}\alpha_i \\ &\quad - b\sum_{i=1}^{n}\alpha_i y_i - \sum_{i=1}^{n}\alpha_i\xi_i - \sum_{i=1}^{n}\beta_i\xi_i \\ &= \sum_{i=1}^{n}\alpha_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j(\boldsymbol{x}_i\cdot\boldsymbol{x}_j) - \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j(\boldsymbol{x}_i\cdot\boldsymbol{x}_j) \\ &\quad + \sum_{i=1}^{n}\xi_i\left(C-\alpha_i-\beta_i\right) - b\sum_{i=1}^{n}\alpha_i y_i.\end{aligned} \tag{25}$$

The last two terms in (25) are actually zero in view of (21)–(22) and the final objective function simplifies to

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j(\boldsymbol{x}_i\cdot\boldsymbol{x}_j). \tag{26}$$

In terms of the constraints, we already imposed $\boldsymbol{\alpha}\in\mathbb{R}^n_+$ and used $\sum_{i=1}^n\alpha_i y_i = 0$. An additional constraint follows from $\alpha_i+\beta_i = C$, see (22). Given that $\beta_i\le 0$, this results in $\alpha_i\le C$. Collecting all the results up to this point, we arrive at the following dual of the SVM Optimization Problem.

SVM Optimization Problem (Dual)

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j(\boldsymbol{x}_i\cdot\boldsymbol{x}_j), \tag{27a}$$

$$\text{subject to:} \quad 0\le\alpha_i\le C \qquad \text{for } i=1,\dots,n \tag{27b}$$

$$\sum_{i=1}^{n}\alpha_i y_i = 0 \tag{27c}$$

The dual defined by (27a)–(27c) is perhaps easier to implement because the only optimization variables are $\boldsymbol{\alpha}$. However, some additional effort is needed to recover $\boldsymbol{w}$ and b. The solution for $\boldsymbol{w}$ follows from $\boldsymbol{w} = \sum_{i=1}^n\alpha_i y_i\boldsymbol{x}_i$. To compute b, we find a support vector. We assume that this support vector has index k. As this support vector is on the marginal hyperplane, it satisfies $y_k(\boldsymbol{w}\cdot\boldsymbol{x}_k+b)=1$ and thus also $\boldsymbol{w}\cdot\boldsymbol{x}_k+b = y_k$.[7] Some rearranging provides

$$b = y_k - \boldsymbol{w}\cdot\boldsymbol{x}_k = y_k - \sum_{i=1}^{n}\alpha_i y_i(\boldsymbol{x}_i\cdot\boldsymbol{x}_k).$$

We have recovered $\boldsymbol{w}$ and b from the solution $\boldsymbol{\alpha}$ and can thus classify new observations.

Remark 6 (Quadratic Program for the Dual Solution of the SVM)
The only solution variables for the dual are the elements of the vector $\boldsymbol{\alpha} = (\alpha_1,\dots,\alpha_n)^\top\in\mathbb{R}^n$. *The*

[7] It suffices to multiply $y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+b)=1$ by y_i. The result follows because $y_i\in\{-1,1\}$ and thus $y_i^2=1$.

constraints are readily implemented with $lb = \mathbf{0}$, $ub = C\boldsymbol{\imath}$, *and* $\boldsymbol{A} = \boldsymbol{y}^\top$, *and* $\boldsymbol{b} = 0$. *As QP solvers are typically implemented as minimizers, we reformulate* (27a) *as*

$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i$$

The linear part of the objective function is constructed as $\boldsymbol{q} = -\boldsymbol{\imath}$. *If we define the matrix* $\boldsymbol{P}$ *with its* $(i,j)^{th}$ *element being equal to* $P_{ij} = y_i \boldsymbol{x}_i \cdot y_j \boldsymbol{x}_j$, *then* $\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i = \frac{1}{2} \boldsymbol{\alpha}^\top \boldsymbol{P} \boldsymbol{\alpha} - \boldsymbol{\imath}^\top \boldsymbol{\alpha}$. *Note that* $\boldsymbol{P} = \text{diag}(\boldsymbol{y}) \boldsymbol{X} \boldsymbol{X}^\top \text{diag}(\boldsymbol{y})$.

2.3 Notes on Implementation

On multiple occasions we have seen how SVMs are solutions of quadratic programs. However, general solvers for quadratic programs do no take into account the particular structure of the SVM's objective function and constraints. Designated packages should be used whenever possible. The following table lists some of the possibilities.

libary	function
pyspark.ml.classification	LinearSVC
sklearn.svm	SVC

3 Applications

`=== Notebook_Applications.ipynb ===`

3.1 MNIST Data Set

The Modified National Institute of Standards and Technology (MNIST) data base is a well-known collection of handwritten digits. This data, first studied in LeCun et al. (2002), can be downloaded from Kaggle.[8] The downloaded data consists of a 60,000 training and 10,000 test images of 28×28 pixels. We apply stratified sampling to the training set to create a validation data set of 10,000 samples. The characteristics of the resulting data sets are summarised in Table 1. The image labels are almost uniformly distributed among the digits.

Figure 4 shows four handwritten images. Each image is an array of grayscale intensities ranging from 0 to 255. While reading the image data, we immediately divide all pixel values by 255 thereby normalising feature values to the unit interval.

3.1.1 Classifying the Digit 8

As a first example we study the SVM that has to decide whether or not an image represents the digit 8. In terms of the notation of Section 2, we assign the class +1 to the images containing a handwritten 8 (and let all other images belong to the class -1). The feature vector $\boldsymbol{x}$ stacks all the normalised pixel values. This vector has length $28 \times 28 = 784$. With a sample size of $50,000$ in the training set, the contribution of $\sum_{i=1}^{n} \xi_i$ can be quite large and small values of C are needed.

[8] At the time of writing the data can from the Kaggle website at this link. An account is required but registration is free of charge.

Table 1: Some summary information regarding the train, validation, and test data set. The columns with the digits from 0 to 9 specify the percentage of observations for the corresponding digit. For example, the 9.87% of the training data images is the handwritten number 0.

data	sample size	0	1	2	3	4	5	6	7	8	9
train	50,000	9.87	11.24	9.93	10.22	9.74	9.04	9.86	10.44	9.75	9.91
valid	10,000	9.87	11.24	9.93	10.22	9.74	9.03	9.86	10.44	9.75	9.92
test	10,000	9.80	11.35	10.32	10.10	9.82	8.92	9.58	10.28	9.74	10.09

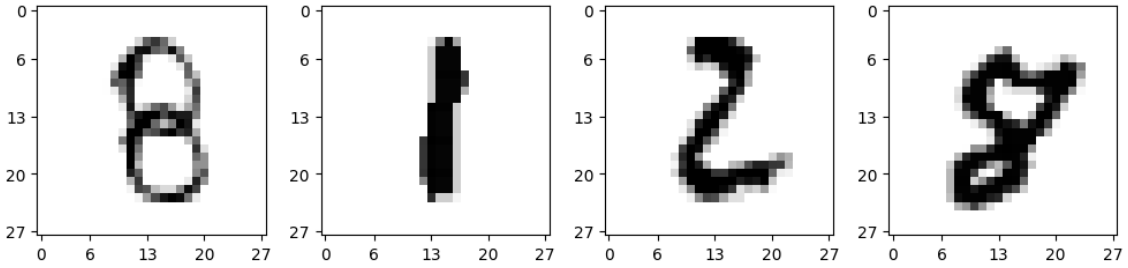

Figure 4: The perfect number 8128 displayed with 28×28 images from the MNIST database. In number theory, a number is considered perfect if the number is equal to the sum of its positive proper divisors. The smallest perfect number is $6 = 1 + 2 + 3$. Verifying whether 8218 is indeed a perfect number is left as an exercise for the reader.

While training, we thus consider a logarithmically equidistant grid on $[10^{-4}, 10^{-1}]$ to explore the hyperparameter space. The results in Figure 5(a) indicate that the classification accuracy on the validation set is rather insensitive to changes in C, whereas computational times differ significantly. As a tradeoff, we decide on $C = 10^{-2}$. The final accuracy on the test data is 96.26%.

For the interpretation of the SVM, we refer to Figure 5(b). We recall that the predicted class is computed as $\hat{y} = \text{sgn}(\hat{b} + \hat{\boldsymbol{w}} \cdot \boldsymbol{x})$. With pixel values having been standardised to the interval $[0, 1]$, positive (negative) parameter values will increase (decrease) the value of $\hat{b} + \hat{\boldsymbol{w}} \cdot \boldsymbol{x}$ and thus help to classify a handwritten digit as $+1$ (-1). The estimated positive parameters, the more yellowish regions in Figure 5(b), indeed resemble the shape of the digit 8. We also note that parameter values around the boundary are close to zero. This is to be expected as these areas are typically left blank in all the input images.

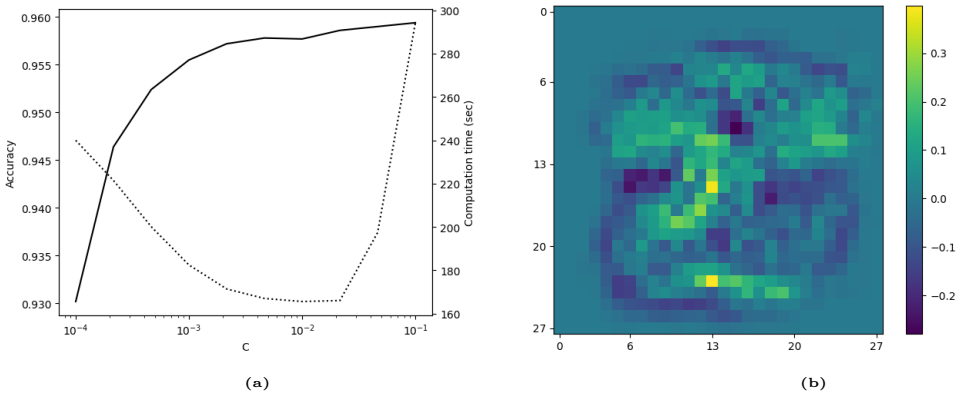

Figure 5: (a) The accuracy (as computed on the validation data) and the computational time when a linear SVM is trained using different values for the hyperparameter C. **(b)** The estimated parameter vector remapped to a 28×28 grid.

References

Anton, H. and C. Rorres (2013). *Elementary Linear Algebra: Applications Version*. John Wiley & Sons.

Boyd, S. P. and L. Vandenberghe (2004). *Convex Optimization*. Cambridge University Press.

Hastie, T., R. Tibshirani, and J. H. Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (2002). Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*(11), 2278–2324.

Mohri, M., A. Rostamizadeh, and A. Talwalkar (2012). *Foundations of Machine Learning*. MIT Press.