Support Vector Machines

Shreeyesh Menon

1 Setup

We consider a binary classification problem with training data

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^n, \qquad \mathbf{x}_i \in \mathbb{R}^d,\ y_i \in \{-1, +1\}.$$

A linear classifier predicts using

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b, \qquad \hat{y} = \operatorname{sign}(f(\mathbf{x})).$$

2 Hard-Margin SVM and Geometric Margin

Assume the data are linearly separable. A separating hyperplane satisfies

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) > 0 \quad \text{for all } i.$$

The *geometric margin* of $(\mathbf{w}, b)$ with respect to $(\mathbf{x}_i, y_i)$ is

$$\gamma_i = \frac{y_i(\mathbf{w}^\top \mathbf{x}_i + b)}{\|\mathbf{w}\|}.$$

The margin of the classifier is the minimum over all points:

$$\gamma = \min_i \gamma_i.$$

We can *rescale* $(\mathbf{w}, b)$ without changing the hyperplane. In particular, we can enforce

$$\min_i y_i(\mathbf{w}^\top \mathbf{x}_i + b) = 1,$$

which implies $\gamma = 1/\|\mathbf{w}\|$. Thus, maximizing the margin is equivalent to minimizing $\|\mathbf{w}\|$. The hard-margin SVM formulation is:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, n. \end{aligned} \tag{P}$$

3 Lagrangian and Dual Derivation (Hard Margin)

We derive the dual of (P). Introduce Lagrange multipliers $\alpha_i \geq 0$ for the inequality constraints $y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 \geq 0$.

The Lagrangian is

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 \right].$$

The primal problem is

$$\min_{\mathbf{w},b} \max_{\boldsymbol{\alpha} \geq 0} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}).$$

The dual problem reverses the order:

$$\max_{\boldsymbol{\alpha} \geq 0} \min_{\mathbf{w},b} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}).$$

Step 1: Minimize $\mathcal{L}$ over $\mathbf{w}$

Compute the derivative w.r.t. $\mathbf{w}$ and set to zero:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i.$$

This shows $\mathbf{w}$ is a linear combination of training examples.

Step 2: Minimize $\mathcal{L}$ over b

Derivative w.r.t. b:

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0.$$

Step 3: Substitute back into the Lagrangian

Plug $\mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$ into $\mathcal{L}$:

$$\begin{aligned}
\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) &= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 \right] \\
&= \frac{1}{2} \left\| \sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j \right\|^2 - \sum_{i=1}^{n} \alpha_i y_i \left(\sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j^\top \mathbf{x}_i + b \right) + \sum_{i=1}^{n} \alpha_i.
\end{aligned}$$

Using $\sum_i \alpha_i y_i = 0$ removes the term involving b. Also,

$$\|\mathbf{w}\|^2 = \left\langle \sum_i \alpha_i y_i \mathbf{x}_i, \sum_j \alpha_j y_j \mathbf{x}_j \right\rangle = \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j.$$

After simplification, the dual function (i.e. the minimized Lagrangian over $\mathbf{w}, b$) becomes

$$g(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j.$$

Hence the dual problem is

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ \text{s.t.} \quad & \alpha_i \geq 0, \quad i = 1, \ldots, n, \\ & \sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned} \tag{D-hard}$$

Decision Function and Support Vectors

At the optimum, we have

$$\mathbf{w}^\star = \sum_{i=1}^{n} \alpha_i^\star y_i \mathbf{x}_i.$$

The prediction function is

$$f(\mathbf{x}) = \mathbf{w}^{\star\top} \mathbf{x} + b^\star = \sum_{i=1}^{n} \alpha_i^\star y_i \mathbf{x}_i^\top \mathbf{x} + b^\star.$$

By the Karush–Kuhn–Tucker (KKT) conditions,

$$\alpha_i^\star \Big(y_i (\mathbf{w}^{\star\top} \mathbf{x}_i + b^\star) - 1 \Big) = 0.$$

Therefore, only points with $\alpha_i^\star > 0$ satisfy $y_i(\mathbf{w}^{\star\top} \mathbf{x}_i + b^\star) = 1$ and lie on the margin: they are the *support vectors*. All other $\alpha_i^\star$ are zero and do not affect the classifier.

4 Soft-Margin SVM and Dual

In non-separable cases, we introduce slack variables $\xi_i \geq 0$ and trade off margin violations via a regularization parameter $C > 0$:

$$\begin{aligned} \min_{\mathbf{w}, b, \boldsymbol{\xi}} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad i = 1, \ldots, n. \end{aligned} \tag{P-soft}$$

The dual (derivation analogous but with extra multipliers for ξ_i) becomes

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ \text{s.t.} \quad & 0 \le \alpha_i \le C, \quad i = 1, \dots, n, \\ & \sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned} \tag{D-soft}$$

The only difference from (D-hard) is the upper bound $\alpha_i \le C$.

5 Kernel Trick

Through some non-linear transformation of the data, we can send the data into a higher-dimensional space where it *becomes* linearly separable. This is called the *kernel trick*.

Replace the dot product $\mathbf{x}_i^\top \mathbf{x}_j$ by any positive semidefinite kernel $K(\mathbf{x}_i, \mathbf{x}_j)$, corresponding to an implicit feature map $\phi(\cdot)$:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j).$$

The kernel effectively applies a nonlinear transformation to the vector x before taking the dot-product.

Then the dual problem uses K instead of $\mathbf{x}_i^\top \mathbf{x}_j$, and the decision function is

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i^\star y_i K(\mathbf{x}_i, \mathbf{x}) + b^\star.$$

6 Loss Function Viewpoint

The soft-margin SVM corresponds to minimizing *hinge loss* plus ℓ_2 regularization:

$$\min_{\mathbf{w}, b} \ \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \max\big(0,\ 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\big).$$

This is a convex optimization problem: the hinge loss is convex, and the regularizer $\|\mathbf{w}\|^2$ is also convex.

7 Relation to the Perceptron

7.1 Perceptron Algorithm

The classical perceptron algorithm seeks a linear classifier that correctly classifies the training data (if such a classifier exists). Starting from $\mathbf{w}_0 = 0$, $b_0 = 0$, it updates whenever a point is misclassified:

$$\text{if } y_i(\mathbf{w}^\top \mathbf{x}_i + b) \le 0 \quad \Rightarrow \quad \mathbf{w} \leftarrow \mathbf{w} + \eta\, y_i \mathbf{x}_i, \qquad b \leftarrow b + \eta\, y_i,$$

where $\eta > 0$ is a learning rate.

One can show that this is equivalent to performing (sub)gradient descent on the *perceptron loss*

$$\ell_{\text{perc}}(y, f(\mathbf{x})) = \max(0, -yf(\mathbf{x})),$$

which only penalizes misclassified points (those with $yf(\mathbf{x}) \leq 0$).

7.2 Similarities between SVM and Perceptron

- Both learn linear decision boundaries of the form $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$.
- Both can be kernelized: the kernel perceptron uses predictions

  $$f(\mathbf{x}) = \sum_{i \in \mathcal{M}} y_i K(\mathbf{x}_i, \mathbf{x}),$$

  where $\mathcal{M}$ is the set of misclassified examples seen so far. This resembles the SVM decision function where we sum over support vectors.
- In the separable case, both algorithms can find a separating hyperplane.

7.3 Key Differences

1. **Margin maximization vs. mere separability.** The perceptron algorithm stops once it finds *some* separating hyperplane. It does not attempt to maximize the margin. In contrast, the hard-margin SVM explicitly maximizes the margin $1/\|\mathbf{w}\|$, which is linked to better generalization.
2. **Loss functions.** The perceptron uses the perceptron loss $\max(0, -yf(\mathbf{x}))$, which only cares about misclassified points. The SVM uses hinge loss $\max(0, 1 - yf(\mathbf{x}))$, which also penalizes *correctly classified* points that are too close to the decision boundary ($0 < yf(\mathbf{x}) < 1$). This encourages a large margin.
3. **Regularization and convex optimization.** The standard SVM formulation includes an explicit ℓ_2 regularizer $\frac{1}{2}\|\mathbf{w}\|^2$, controlling model complexity. The SVM objective is convex, and its global optimum can be found using quadratic programming. The classical perceptron update is an online algorithm without explicit regularization; it does not solve a fixed convex optimization problem.
4. **Support vectors vs. all updates.** In SVMs, only support vectors (points with $\alpha_i^\star > 0$) affect the final classifier. For the perceptron, $\mathbf{w}$ is the sum of updates from *all* misclassified examples encountered during training.