Support Vector Machine

Bùi Tiến Lên

2024

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Contents

Notation

symbol	meaning
$a, b, c, N \dots$	scalar number
$\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{x}, \boldsymbol{y} \dots$	column vector
$\boldsymbol{X}, \boldsymbol{Y} \dots$	matrix
$\mathbb{R}$	set of real numbers
$\mathbb{Z}$	set of integer numbers
$\mathbb{N}$	set of natural numbers
$\mathbb{R}^D$	set of vectors
$\mathcal{X}, \mathcal{Y}, \dots$	set
$\mathcal{A}$	algorithm

operator	meaning
$\boldsymbol{w}^\intercal$	transpose
$\boldsymbol{X}\boldsymbol{Y}$	matrix multiplication
$\boldsymbol{X}^{-1}$	inverse
$\boldsymbol{x} \cdot \boldsymbol{y}$	dot

Linear Support Vector Machines

- The Separable Case
- The Non-Separable Case

Problem Statement

- Training set:

$$(\boldsymbol{x}_i, y_i)_{i=1\dots N} \in \mathbb{R}^D \times \{-1, 1\}$$

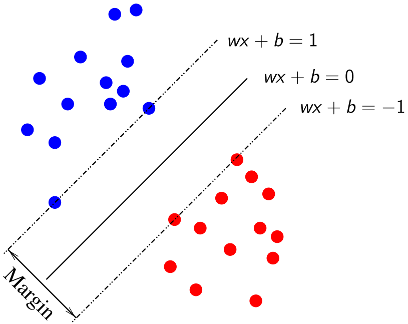
- We would like to find an hyperplane

$$\boldsymbol{w}^\intercal \boldsymbol{x} + b = 0, \quad (\boldsymbol{w} \in \mathbb{R}^D, b \in \mathbb{R})$$

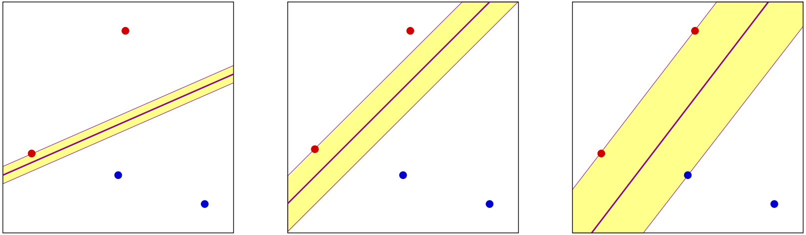
which **separates** the two classes.

Margin

- Let d_+ be the shortest distance from the hyperplane to the closest positive example.
- Let d_- be the shortest distance from the hyperplane to the closest negative example.
- Define the **margin** of the hyperplane to be $\min(d_+, d_-)$.



Better Linear Separation

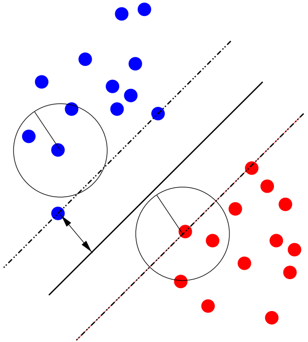


Two questions:

1. Why is bigger margin better?
2. Which $\boldsymbol{w}$, b maximizes the margin

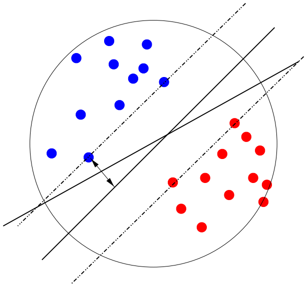
Why is it Good to Maximize the Margin?

- If training and test data come from the same distribution and all test data are within some Δ distance from the training points. If all points lie at a distance of at least Δ from the separator, and all points are in a bounded sphere, then a small perturbation of the definition of the separator will not hurt.



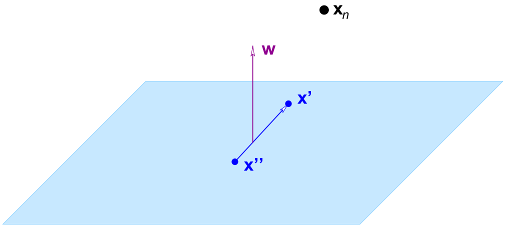
Why is it Good to Maximize the Margin? (cont.)

- One can use less bits to encode the separating hyperplane (**Minimum Description Length** principle)



Finding w with large margin

- Let $\mathbf{x}_n$ be the nearest data point to the plane $\mathbf{w}^\intercal \mathbf{x} + b = 0$. How far is it?



- The distance between $\mathbf{x}_n$ and the plane $\mathbf{w}^\intercal \mathbf{x} + b = 0$ where $|\mathbf{w}^\intercal \mathbf{x}_n + b| = 1$ (normalize $\mathbf{w}$ and b)

$$distance = \frac{1}{|\mathbf{w}|} \tag{1}$$

A Constrained Optimization Problem

- Representation of hypothesis set

$$\mathcal{H} : y = f(\boldsymbol{x}) = \text{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b) \tag{2}$$

- Evaluation

$$\begin{aligned} \arg\min_{\boldsymbol{w},b} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 && (3) \\ \text{subject to} \quad & y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1, \quad i = 1, 2, \ldots, N && (4) \end{aligned}$$

The Dual Formulation

- Representation of hypothesis set

$$\mathcal{H} : y = f(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i (\boldsymbol{x} \cdot \boldsymbol{x}_i) + b\right) \tag{5}$$

- Evaluation

$$\arg\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) - \sum_{i=1}^{N} \alpha_i \tag{6}$$

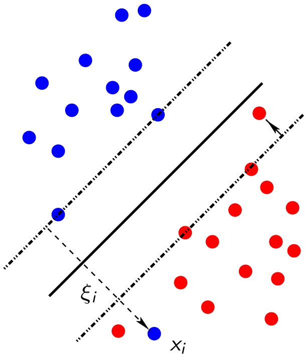
$$\text{subject to} \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{7}$$

$$\alpha_i \geq 0, \quad i = 1, 2, \ldots, N \tag{8}$$

- This can be solved using classical **quadratic programming optimization**

Bug

- This minimization problem does not have any solution if the two classes are not separable.



Fixing The Bug: "Soft" Margin

- Relax the constraints: use a **soft margin** instead of a **hard margin**.
- We would like to minimize:

$$\arg\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i \tag{9}$$

$$\text{subject to} \quad y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \geq 1 - \xi_i \tag{10}$$

$$\xi_i \geq 0, \quad i = 1, 2, \ldots, N \tag{11}$$

The Dual Formulation

- Representation of hypothesis set

$$\mathcal{H} : y = f(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i (\boldsymbol{x} \cdot \boldsymbol{x}_i) + b\right) \tag{12}$$

- Evaluation

$$\arg\min_{\boldsymbol{\alpha}} \quad \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) - \sum_{i=1}^{N} \alpha_i \tag{13}$$

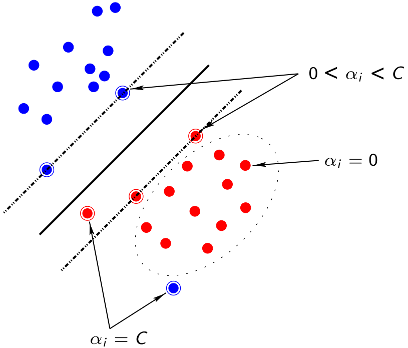
$$\text{subject to} \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{14}$$

$$0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, N \tag{15}$$

Support Vector Terminology

- Training examples $\boldsymbol{x}_i$ with $\alpha_i > 0$ are **support vectors**.

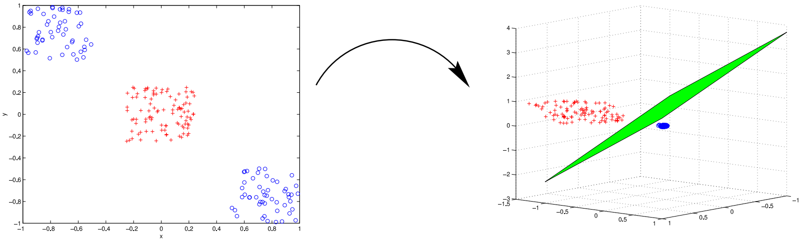
$$\begin{aligned} \alpha_i = 0 &\Rightarrow y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) > 1 \\ \alpha_i = C &\Rightarrow y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) < 1 \\ 0 < \alpha_i < C &\Rightarrow y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) = 1 \end{aligned} \tag{16}$$



Kernels Support Vector Machines

Non-Linear SVMs

- Project the data into a **higher dimensional space** (**feature space**): it should be easier to separate the two classes.
- Given a function $\phi : \mathbb{R}^D \rightarrow \mathcal{F}$, work with $\phi(\boldsymbol{x}_i)$ instead of working with $\boldsymbol{x}_i$.



The Kernel Function

Concept 1

A **kernel** is a function $k(\boldsymbol{x}, \boldsymbol{z})$ which represents a dot product in a "hidden" feature space of ϕ.

$$k(\boldsymbol{x}, \boldsymbol{z}) = \phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{z}) \tag{17}$$

- **Note that**: we have only dot products $\phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j)$ to compute; however, this could be very expensive in a high dimensional space.
- **Kernel trick**:

$$\text{instead of } \phi(\boldsymbol{x}) = \phi\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}, \text{ use } k(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x} \cdot \boldsymbol{z})^2$$

Common Kernels

- Polynomial:

$$k(\boldsymbol{x}, \boldsymbol{z}) = (u\boldsymbol{x} \cdot \boldsymbol{z} + v)^p \ (u \in \mathbb{R}, v \in \mathbb{R}, p \in \mathbb{N}) \tag{18}$$

- Gaussian:

$$k(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{z}\|^2}{\sigma^2}\right), \sigma \in \mathbb{R}^+ \tag{19}$$

Techniques for Construction of Kernels

In all the following, $k_1, k_2, ..., k_j$ are assumed to be valid kernel functions

1. **Scalar multiplication**: The validity of a kernel is conserved after multiplication by a positive scalar, i.e., for any $\alpha > 0$, the function

$$k(\boldsymbol{x}, \boldsymbol{z}) = \alpha k_1(\boldsymbol{x}, \boldsymbol{z}) \tag{20}$$

2. **Adding a positive constant**: For any positive constant $\alpha > 0$, the function

$$k(\boldsymbol{x}, \boldsymbol{z}) = \alpha + k_1(\boldsymbol{x}, \boldsymbol{z}) \tag{21}$$

Techniques for Construction of Kernels (cont.)

3. **Linear combination**: A linear combination of kernel functions involving only positive weights, i.e.,

$$k(\boldsymbol{x}, \boldsymbol{z}) = \sum_{j=1}^{m} \alpha_j k_j(\boldsymbol{x}, \boldsymbol{z}), \qquad \text{with } \alpha_j > 0 \tag{22}$$

   is a valid kernel function.

4. **Product**: The product of two kernel functions, i.e.,

$$k(\boldsymbol{x}, \boldsymbol{z}) = k_1(\boldsymbol{x}, \boldsymbol{z}) k_2(\boldsymbol{x}, \boldsymbol{z}) \tag{23}$$

   is a valid kernel function.

Techniques for Construction of Kernels (cont.)

5. **Polynomial functions of a kernel output**: Given a polynomial $f : \mathbb{R} \to \mathbb{R}$ with positive coefficients, the function

$$k(\boldsymbol{x}, \boldsymbol{z}) = f(k_1(\boldsymbol{x}, \boldsymbol{z})) \tag{24}$$

is a valid kernel function.

6. **Exponential function of a kernel output**: The function

$$k(\boldsymbol{x}, \boldsymbol{z}) = \exp(k_1(\boldsymbol{x}, \boldsymbol{z})) \tag{25}$$

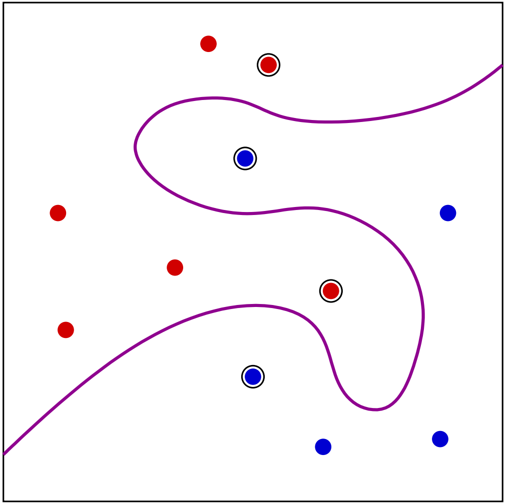
is a valid kernel function.

7. **Product of matrix** and **vectors**:

$$k(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{x}^{\mathsf{T}} A \boldsymbol{z} \tag{26}$$

where A is a symmetric positive semidefinite matrix.

Decision Boundary and Support Vectors



SVMs in Practice

- In order to tune the **capacity**, the kernel is the most important parameter to choose.
  - Polynomial kernel: increasing the degree will increase the **capacity**.
  - Gaussian kernel: increasing σ will decrease the capacity.
- Tune C, the trade-off between the **margin** and the **errors**.
  - For non-noisy data sets, C usually has not much influence.
  - Carefully choose C for noisy data sets: small values usually give better results.

Multi-class SVM

Multiclass SVM formulations

- There are a few ways of formulating the SVM over multiple classes:
  - One-vs-all
  - One-vs-one
  - Hierarchical
  - **Multiclass**

Score function

Concept 2

The **score function** f that maps the raw features to class scores.

$$\boldsymbol{z} = f(\boldsymbol{x}; W) = W\boldsymbol{x} \tag{27}$$

input image

$$\underbrace{\begin{bmatrix} 1.1 & 0.2 & -0.5 & 0.1 & 2.0 \\ 3.2 & 1.5 & 1.3 & 2.1 & 0.0 \\ -1.2 & 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{W} \times \underbrace{\begin{bmatrix} 1 \\ 56 \\ 231 \\ 24 \\ 2 \end{bmatrix}}_{\boldsymbol{x}} = \begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix} \begin{array}{l} \text{cat score} \\ \text{dog score} \\ \text{ship score} \end{array}$$

Multiclass SVM loss

- Given the input vector $\boldsymbol{x}_i$ and the label y_i that specifies the index of the correct class. The multiclass SVM loss (**hinge loss**) for the vector $\boldsymbol{x}_i$ is then formalized as follows

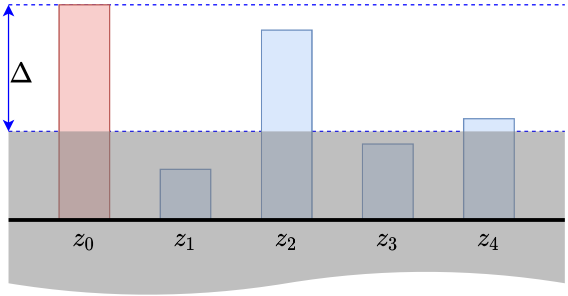
$$L_i = \sum_{j \neq y_i} \max(0, z_j - z_{y_i} + \Delta) \tag{28}$$

where $\boldsymbol{z} = f(\boldsymbol{x}_i; W) = W\boldsymbol{x}_i$

Example

- Suppose that we have five classes $\{0, 1, 2, 3, 4\}$ that receive the scores $\boldsymbol{z} = [17, 4, 15, 6, 8]$ and the true class $y_i = 0$
- Also assume that $\Delta = 10$

$$L_i = \max(0, 4 - 17 + 10) + \max(0, 15 - 17 + 10) + \max(0, 6 - 17 + 10) + \max(0, 8 - 17 + 10) = 9$$



Regularization loss

- The most common regularization penalty is the L_2 norm that discourages large weights through an elementwise quadratic penalty over all parameters:

$$R(W) = \sum_k \sum_l W_{k,l}^2 \tag{29}$$

- The data loss (which is the average loss L_i over all examples) and the regularization loss. That is, the full multiclass SVM loss becomes:

$$\mathcal{L} = \mathcal{L}_{\text{data}} + \mathcal{L}_{\text{reg}} = \underbrace{\frac{1}{N}\sum_i L_i}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}} \tag{30}$$

- **Learning goal**: Find W that minimize

$$\arg\min_W \mathcal{L} \tag{31}$$

Practical considerations

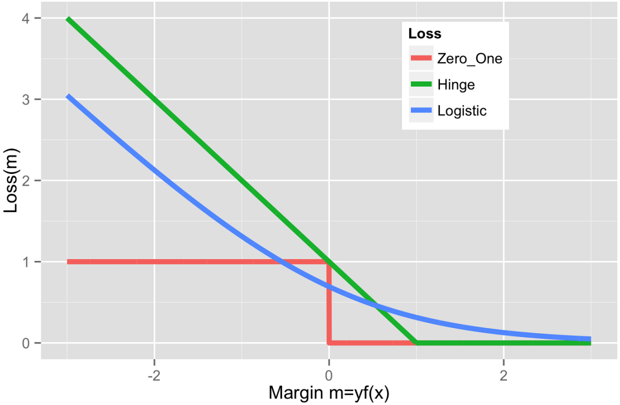
- **Setting Delta**: It can safely be set to $\Delta = 1.0$ in all cases
- **Relation to Binary Support Vector Machine**: The loss for the i-th example $(\boldsymbol{x}_i, y_i)$ can be written as

$$L_i = C \max(0, 1 - y_i \boldsymbol{w}^\top \boldsymbol{x}_i) + R(\boldsymbol{w}) \tag{32}$$

where C is a hyperparameter, and $y_i \in \{-1, 1\}$

Binary classification losses

- Perceptron (zero-one)
- SVM (hinge)
- Logistic



SGD for hinge loss

- Consider linear hypothesis space:

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}^\intercal \boldsymbol{x} \tag{33}$$

- Hinge loss of $(\boldsymbol{x}, y)$

$$\mathcal{L}(x) = \max(0, 1 - y\boldsymbol{w}^\intercal \boldsymbol{x}) \tag{34}$$

- Gradient of hinge loss $(\boldsymbol{x}, y)$:

$$\nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{x}) = \begin{cases} -y\boldsymbol{x} & \text{if } yh_{\boldsymbol{w}}(\boldsymbol{x}) < 1 \\ 0 & \text{if } yh_{\boldsymbol{w}}(\boldsymbol{x}) > 1 \\ \text{undefined} & \text{if } yh_{\boldsymbol{w}}(\boldsymbol{x}) = 1 \end{cases} \tag{35}$$

- A point with margin $m = yh_{\boldsymbol{w}}(\boldsymbol{x}) = 1$ is correctly classified $\rightarrow$ we can skip SGD update for these points.

References

Goodfellow, I., Bengio, Y., and Courville, A. (2016).
Deep learning.
MIT press.

Lê, B. and Tô, V. (2014).
Cở sở trí tuệ nhân tạo.
Nhà xuất bản Khoa học và Kỹ thuật.

Russell, S. and Norvig, P. (2021).
Artificial intelligence: a modern approach.
Pearson Education Limited.