西交利物浦大学
Xi'an Jiaotong-Liverpool University

Pattern Recognition

Lecture 15. Support Vector Machine

Dr. Shanshan ZHAO

School of AI and Advanced Computing
Xi'an Jiaotong-Liverpool University

Academic Year 2023-2024

Table of Contents

Outline

Introduction

- In the general case, the problem of finding linear discriminant functions can be formulated as a problem of **optimizing a criterion function**.
- Among all hyperplanes separating the data, there exists **a unique one yielding the maximum margin** of separation between the classes.

Binary Classification

Given training data (x_i, y_i) for $i = 1 \ldots N$, with $x_i \in \mathcal{R}^d$ and $y_i \in \{-1, 1\}$, learn a classifier $f(x)$ such that

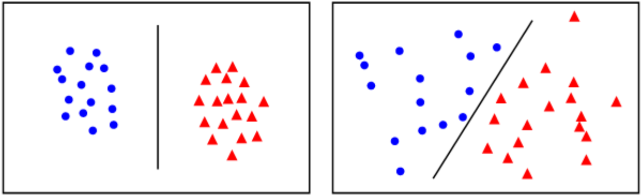
$$f(x_i) \begin{cases} \geq 0 & y_i = +1 \\ < 0 & y_i = -1 \end{cases}$$

i.e.

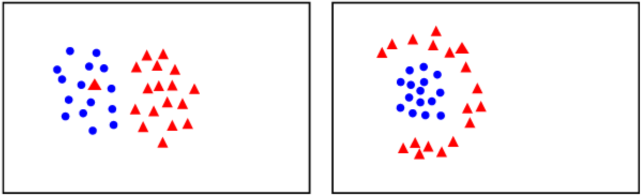
$y_i f(x_i) > 0 \rightarrow$ a correction classification

Linear separability

linearly separable



not linearly separable

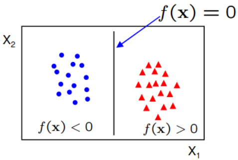


Linear Classifiers

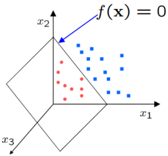
A linear classifier has the form

$$f(x) = w^T x + b$$

- w is normal(vertical) to the line, and the b is the bias/intercept
  - *whether the positive of f(x) is on the right or left of the line depends on the sign of the first parameter in vector w.*
- w is known as the weight vector.



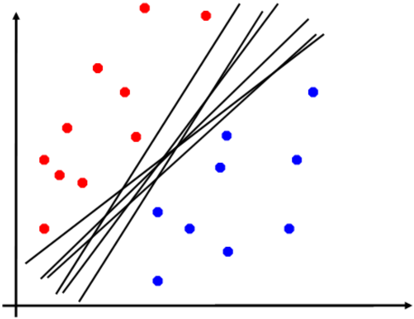
(a) a line in 2D



(b) a plane in 3D

Linear Classifiers

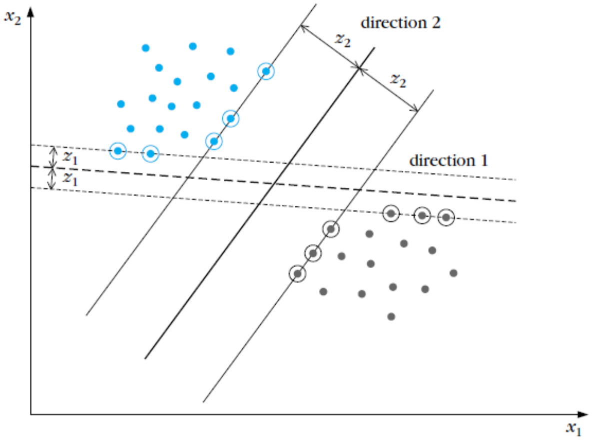
- If training data is linearly separable, perceptron is guaranteed to find some linear separator/decision hyperplane.
- Which of these is optimal?



Outline

SVM Intuition

a very sensible choice for the hyperplane classifier would be the one that leaves the maximum margin from both classes.



SVM

margin: a hyperplane leaves from both classes.
Our goal is to search for the direction that gives the maximum possible margin.
Recall that the distance of a point from a hyperplane is given by

$$z = \frac{|g(x)|}{||w||}$$

We can scale w, b so that the value of $g(x)$, at the nearest points in c_1, c_2 (circled in figure).

SVM

We can scale w, w_0 so that the value of $g(x)$, at the nearest points in c_1, c_2 (circled in figure1), is equal to 1 for class c_1 and equal to -1 for class c_2, which is equivalent with

- 1. Having a margin of $\frac{1}{||w||} + \frac{1}{||w||} = \frac{2}{||w||}$
- 2. Requiring that

$$\begin{cases} w^T x + b \geq 1, & \forall x \in c_1 \\ w^T x + b \leq -1, & \forall x \in c_2 \end{cases}$$

- The support vectors lie on either of the two hyperplanes, that is

$$w^T x + b = \pm 1$$

Objective: Maximizing the margins

Maximize the margin (I) –Primal form(*)

- Optimization (Quadratic Programming) (known as a **Primal problem**.

$$\begin{cases} \text{minimize} \quad J(w,b) = \frac{1}{2}||w||^2 \\ \text{subject to} \quad y_i(w^T x_i + b) \geq 1, \quad i = 1,2,...,N \end{cases} \tag{1}$$

- Minimizing the norm makes the margin maximum

It belongs to the convex programming family of problems, since the cost function is convex and the constraints are linear and define a convex set of feasible solutions. Such problems can be solved by considering the socalled Lagrangian duality.

Maximize the margin (II) –Dual form(*)

- The objective in Eq. (1) is a standard quadratic programming problem.
- Let $\lambda \in \mathscr{R}^N$ be the dual variables, corresponding to Lagrange multipliers that enforce the N inequality constraints.

The generalized Lagrangian is given below

$$\mathscr{L}(w, b, \lambda) = \frac{1}{2}w^T w - \sum_{i=1}^{N} \lambda_i [y_i(w^T x_i + b) - 1] \tag{2}$$

where λ is the Lagrange multiplier

Maximize the margin (II) –Dual form(*)

By setting each partial derivative equal to zero, We can obtain the parameters (coefficients) of the hyperplane from the Lagrange multipliers

$$\frac{\partial}{\partial w}\mathscr{L}(w,b,\lambda) = 0 \tag{3}$$

$$\frac{\partial}{\partial b}\mathscr{L}(w,b,\lambda) = 0 \tag{4}$$

$$\lambda_i \geq 0, \quad i = 1,2,...,N \tag{5}$$

$$\lambda_i[y_i(w^T x_i + b) - 1] = 0, \quad i = 1,2,...,N \tag{6}$$

Combining (3) (4) and (2), results in

$$w = \sum_{i=1}^{N} \lambda_i y_i x_i \tag{7}$$

$$\sum_{i=1}^{N} \lambda_i y_i = 0 \tag{8}$$

where x_i belongs to support vectors, $y_i \in \{1,-1\}$

Maximize the margin (II) –Dual form(*)

Plugging these into Lagrangian yields the following

$$\mathscr{L}(w, b, \lambda) = \frac{1}{2}w^T w - \sum_{i=1}^{N} \lambda_i y_i w^T x_i - \sum_{i=1}^{N} \lambda_i y_i b + \sum_{i=1}^{N} \lambda_i$$

$$= \frac{1}{2}w^T w - w^T w - 0 + \sum_{i=1}^{N} \lambda_i$$

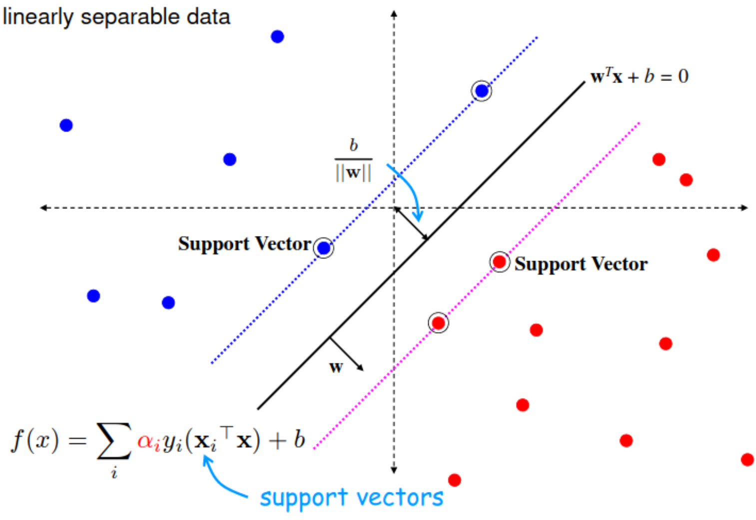
$$= -\frac{1}{2}\sum_{i=1}^{N}\sum_{i=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_i + \sum_{n=1}^{N} \lambda_i$$

After a long process, we but a numerically stable solution for b

$$b = \frac{1}{|\mathcal{S}|}\sum_{i \in \mathcal{S}}(y_i - w^T x_i) = \frac{1}{|\mathcal{S}|}\sum_{i \in \mathcal{S}}(y_i - \sum_{j \in \mathcal{S}} \lambda_j y_j x_j^T x_i) \quad (9)$$

where $\mathcal{S}$ is the set of support vectors.

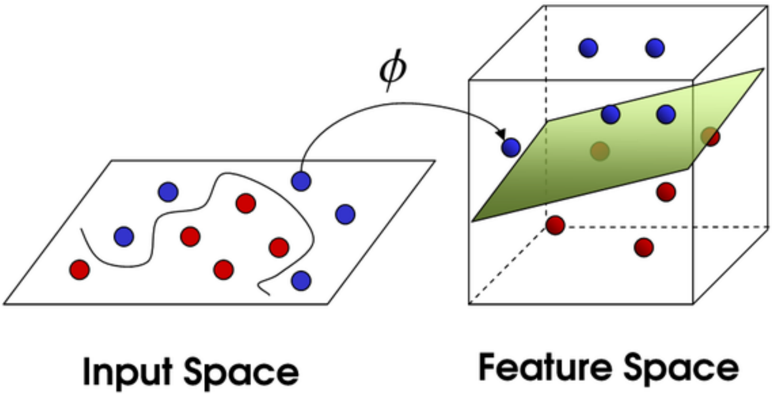
SVM


linearly separable data
$\mathbf{w}^T\mathbf{x} + b = 0$
$\frac{b}{||\mathbf{w}||}$
Support Vector
Support Vector
$\mathbf{w}$
$f(x) = \sum_i \alpha_i y_i (\mathbf{x}_i{}^\top \mathbf{x}) + b$
support vectors

SVM Training Methodology

- Training is formulated as an optimization problem
  - Dual problem - Makes use of Lagrange multipliers
  - Kernel trick
- Determination of the model parameters corresponds to a convex optimization problem
  - Solution is straightforward (local solution is a global optimum)

Kernel trick : Feature mapping



SVM : Kernel trick

- Rather than applying SVMs using the original input attributes x, we may instead want to learn using some features $\phi(x)$.
- To do so, we simply need to go over our previous algorithm, and replace x everywhere in it with $\phi(x)$.
- Since the algorithm can be written entirely in terms of the inner products $\langle x, z \rangle$, this means that we would replace all those inner products with$\langle \phi(x), \phi(z) \rangle$.

SVM : Kernel trick

- Both the quadratic programming problem and the final decision function

$$g(x) = \text{sign}(\sum_{i=1}^{n} \lambda_i y_i \langle x \cdot x_i \rangle + b) \tag{10}$$

depend only on the dot procucts between patterns

- We can generalize this result to the non-linear case by mapping the original input space into some other space $\mathscr{F}$ using a non-linear map $\phi : \mathscr{R}^d \to \mathscr{F}$ and perform the linear algorithm in the $\mathscr{F}$ space which only requires the dot products

$$k(x, y) = \phi(x)\phi(y)$$

SVM : Kernel trick

- Even though $\mathscr{F}$ may be high-dimensional, a simple kernel $k(x, y)$ such as the following can be computed efficiently.

Polynomial	$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^p$
Sigmoidal	$k(\mathbf{x}, \mathbf{y}) = \tanh(\kappa(\mathbf{x} \cdot \mathbf{y}) + \theta)$
Radial basis function	$k(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2 / (2\sigma^2))$

Figure 2: Common kernel functions

- Once a kernel function is chosen, we can substitute $\phi(x_i)$ for each training example x_i, and perform the optimal hyperplane algorithm in $\mathscr{F}$.

SVM : Kernel trick

- This results in the non-linear decision function of the form

$$g(x) = \mathrm{sign}(\sum_{i=1}^{n} \lambda_i y_i k(x, x_i) + b) \tag{11}$$

  where the parameters λ_i are computed as the solution of the quadratic programming problem.
- In the original input space, the hyperplane corresponds to a non-linear decision function whose form is determined by the kernel.
- **We can use $k(x, x')$ directly for computation without transforming x and x', as long as z exists.**

Kernel : inner product

Polynomial kernels

$\forall x, x' \in \mathscr{R}^2$,

$$k(x, x') = (1 + x^T x')^2 = (1 + x_1 x_1' + x_2 x_2')^2$$

$$= 1 + x_1^2 x_1'^2 + x_2^2 x_2'^2 + 2x_1 x_1' + 2x_2 x_2' + 2x_1 x_1' x_2 x_2'$$

$\Downarrow$ above is an inner product of two vectors

$$= \begin{bmatrix} 1 \\ x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \sqrt{2}x_1 x_2 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ x_1'^2 \\ x_2'^2 \\ \sqrt{2}x_1' \\ \sqrt{2}x_2' \\ \sqrt{2}x_1' x_2' \end{bmatrix}$$

$$= z^T z'$$

with $z = \phi(x) = [1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2]$

Example

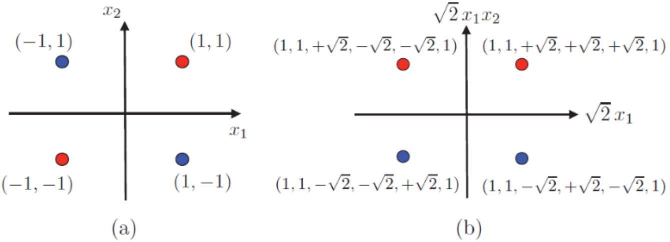


Figure 5.2 Illustration of the XOR classification problem and the use of polynomial kernels. (a) XOR problem linearly non-separable in the input space. (b) Linearly separable using second-degree polynomial kernel.

non-separable class case

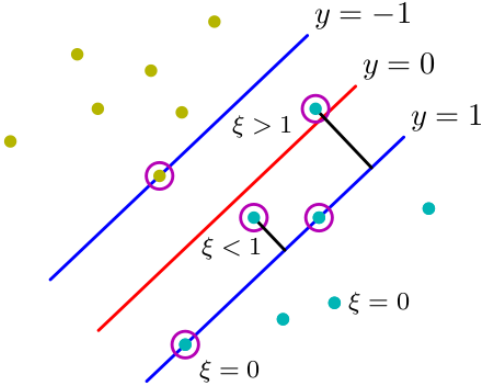
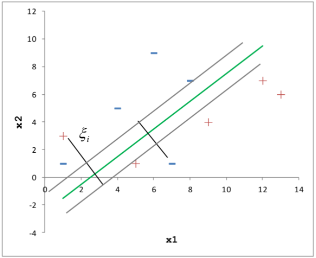


Figure 3: Illustration of the slack variables $\xi_i \geq 0$. Data points with circles around them are support vectors.

Soft Margin

$$y_i(w^T x_i + b) \geq 1 - \xi_i, i = 1, 2, ..., N$$

- ξ is a vector of size n
- $\xi_i \geq 0$ marks the misclassified instances
- $\xi_i = 0$, the instance is in the right side of the margin
- $\xi_i < 1$, the instance is in the right side of the maximum margin hyperplane, but it exceeds its 0 margin
- $\xi_i > 1$, the instance is misclassified i.e. it is in the wrong side of the maximum



Soft Margin

Using the slack variables ξ_i to handle misclassified instances.

- The new optimization problems becomes:

$$\begin{cases} \text{minimize} & \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} & \begin{cases} w^T x_i + b \geq 1 - \xi_i & \text{for} \quad y_i = +1, \\ w^T x_i + b \leq -1 + \xi_i & \text{for} \quad y_i = -1, \\ \xi_i \geq 0 & i = 1, 2, ..., n \end{cases} \end{cases} \tag{12}$$

  - Where $\xi_i, i = 1, 2, ..., n$, are called the slack variables and C is a regularization parameter.
  - The term $C\sum_{i=1}^{n}\xi_i$ can be thought of as measuring some amount of misclassification where lowering the value of C corresponds to a smaller penalty for misclassification.

This is still a quadratic optimization problem and there is a unique minimum.

SVM

- Maximize the margin (I) –Primal form
- Maximize the margin (II) –Dual form
- Nonlinear classification – Kernel trick
- Noisy labels – Soft Margin

Check the code.

Thank You !

$\mathscr{Q}$ & $\mathscr{A}$