Support Vector Machine

Sanjay Singh[⋆†]

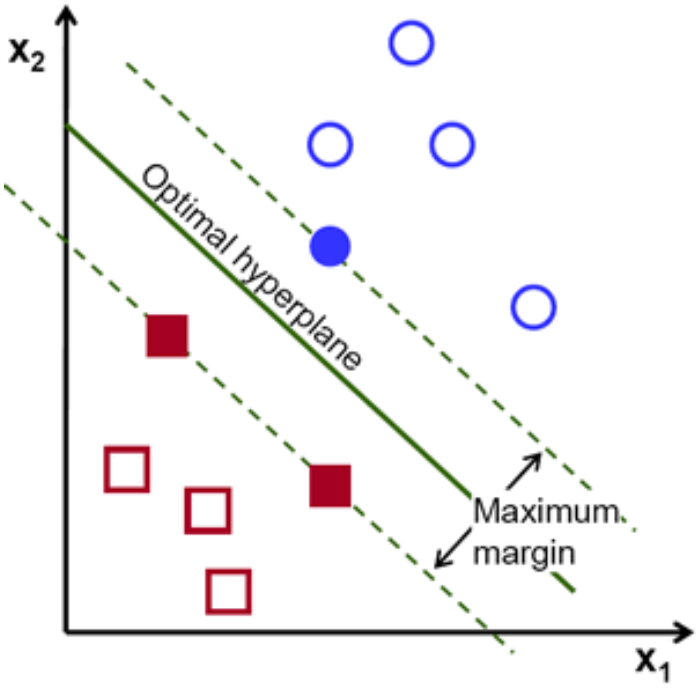
[⋆]Department of Information and Communication Technology
Manipal Institute of Technology, Manipal University
Manipal-576104, INDIA
sanjay.singh@manipal.edu

[†]Centre for Artificial and Machine Intelligence (CAMI)
Manipal University, Manipal-576104, INDIA

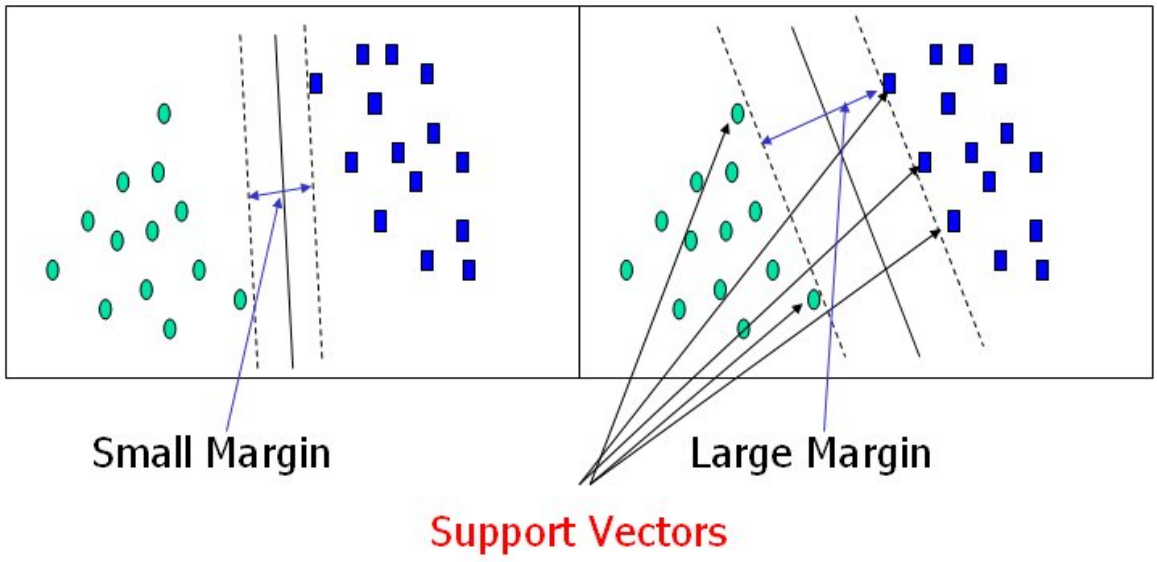
March 14, 2019

Main Idea

Given a training sample, the support vector machine (SVM) constructs a hyperplane as the decision surface in such a way that the margin of separation between positive and negative examples is maximized.



- **Support vectors**:input points closest to the separating hyperplane
- **Margin of separation** ρ: distance between the separating hyperplane and the closest input point



Introduction

- Support Vector Machine (SVM) is a linear machine with some nice properties
- Basic idea of SVM is to construct a separating hyperplane where the margin of separation between positive and negative examples are maximized
- Principled derivation: structural risk minimization
  - Error rate is bounded by
    - training error-rate, and
    - VC-dimension of the model
  - SVM makes training error-rate zero and minimizes the VC-dimension of the model
- SVM provides good generalization performance without incorporating problem-domain knowledge

- Inner-product kernel between support vector x_i and vector x drawn from the input space is central to the construction of SVM
- Depending on how inner-product kernel is generated, we may construct different learning machines characterized by nonlinear decision surfaces of their own
- We'll use support vector learning algorithm to construct following learning machines:
  - Polynomial learning machines
  - Radial-basis function networks
  - Two-layer perceptron
- For each of these feedforward networks we'll use support vector learning algorithm to implement the learning process
- Support vector learning algorithm is of generic nature

Some Definitions:Linear Optimization

- Linear optimization (aka linear programming) is method to achieve optimal value in a mathematical model where requirements are represented by linear equation
- Linear programming is a special case of mathematical programming

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & C^T X \\ \text{subject to} \quad & AX \leq b \\ & X \geq 0 \end{aligned}$$

- Quadratic optimization is the problem of optimizing a quadratic function subjected to linear constraints on variables

Some Definitions: Primal and Dual Problem

- If the primal problem is

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

- The dual problem is defined as

$$\begin{aligned} \underset{y}{\text{minimize}} \quad & b^T y \\ \text{subject to} \quad & A^T y \geq c \end{aligned}$$

- If x is feasible for the primal and y is feasible for the dual then

$$c^T x \leq b^T y$$

- So $(\text{primal optimal}) \leq (\text{dual optimal})$

Some Definitions:Lagrange Multiplier

- In optimization theory, the method of Lagrange multiplier is a strategy for finding the local minima or maxim of a function subjected to equality constraint
- Consider the optimization problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g(x) = c \end{aligned}$$

- Both f and g must be continuous function
- Let a new variable λ called a Lagrange multiplier, and define the Lagrange function as

$$\Lambda(x, \lambda) = f(x) + \lambda(g(x) - c)$$

- The λ term may be either added or subtracted
- If $f(x_0)$ is a minimum of f(x) for the original constraint problem then there exist λ_0 such that (x_0, λ_0) is a stationary point

Example on Lagrange Multiplier

- Let $f(x, y) = x^2 - 8x + y^2 - 12y + 48$
-

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x, y) \\ \text{subject to} \quad & x + y = 8 \end{aligned}$$

- Lagrange function can be written as

$$F(x, y, \lambda) = x^2 - 8x + y^2 - 12y + 48 + \lambda(x + y - 8)$$

- Set the partial derivative of the function equal to zero
- $\frac{\partial F}{\partial x} = 2x - 8 + \lambda = 0$
- $\frac{\partial F}{\partial y} = 2y - 12 + \lambda = 0$
- $\frac{\partial F}{\partial \lambda} = x + y - 8 = 0$
- On solving these equations we get $(x, y) = (3, 5)$

- $f(x, y, z) = xy + yz$
- $$\begin{aligned} \underset{x}{\text{maximize}} \quad & f(x, y, z) \\ \text{subject to} \quad & x + 2y = 6 \\ & x = 3z \end{aligned}$$
- Lagrange function can be written as

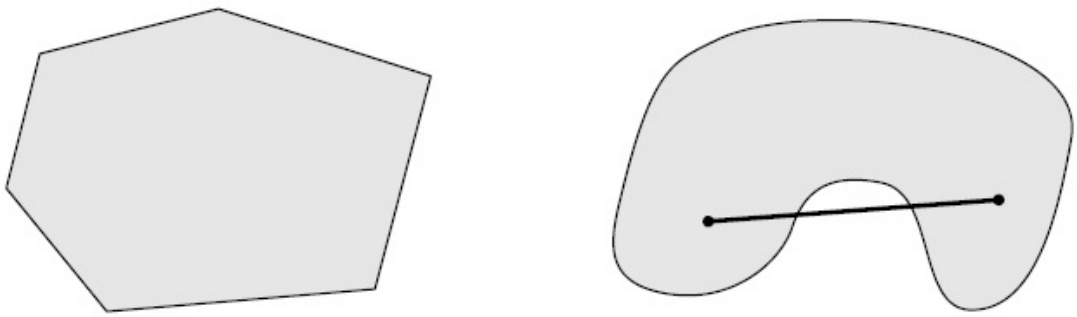
$$F(x, y, z, \lambda, \mu) = xy + yz + \lambda(x + 2y - 6) + \mu(x - 3z)$$

Convex Set

Definition

A set C is convex if, for any $x, y \in C$ and $\theta \in \mathbb{R}$ with $0 \leq \theta \leq 1$,

$$\theta x + (1 - \theta)y \in C$$



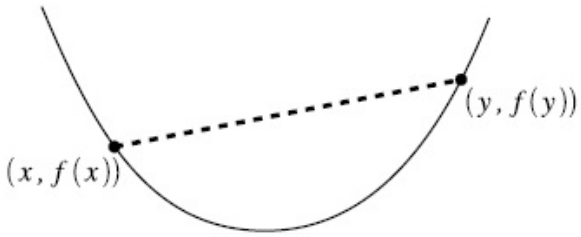
Examples of a convex set (a) and a non-convex set (b)

Convex Function

Definition

A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if its domain (denoted $\mathcal{D}(f)$) is a convex set, and if, for all $x, y \in \mathcal{D}(f)$ and $\theta \in \mathbb{R}$ with $0 \leq \theta \leq 1$,

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y).$$



Graph of a convex function

- Function is **strictly convex** if definition holds with strict inequality for $x \neq y$ and $0 \leq \theta \leq 1$
- We say f is **concave** if $-f$ is convex
- f is strictly concave if $-f$ is strictly convex

First Order Condition for Convexity

Suppose a function $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable (i.e., the gradient $\nabla_x f(x)$ exists at all points $x \in \mathcal{D}(f)$), then f is convex iff $\mathcal{D}(f)$ is a convex set and for all $x, y \in \mathcal{D}(f)$,

$$f(y) \geq f(x) + \nabla_x f(x)^T (y - x)$$

Second Order Condition for Convexity

Suppose a function $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable (i.e., the Hessian $\nabla_x^2 f(x)$ exists at all points $x \in \mathcal{D}(f)$), then f is convex iff $\mathcal{D}(f)$ is a convex set and its Hessian is positive semidefinite: i.e., for any $x \in \mathcal{D}(f)$,

$$\nabla_x^2 f(x) \succeq 0.$$

(Here, the notation $\succeq$ when used in conjunction with matrices refers to positive semidefinitiness, rather than componentwise inequality). In one dimension, this is equivalent to the condition that $f''(x)$ always be non-negative.

Examples of Convex Functions

- **Exponential.** Let $f : \mathbb{R} \to \mathbb{R}, f(x) = e^{ax}$ for any $a \in \mathbb{R}$. To show $f(x)$ is convex, $f''(x) = a^2 e^{ax}$, which is positive for all x
- **Negative logarithm.** Let $f : \mathbb{R} \to \mathbb{R}, f(x) = -\log x$ with domain $\mathcal{D}(f) = \mathbb{R}_{++}$, (here, $\mathbb{R}_{++}$ denotes the set of strictly positive real numbers, $\{x : x > 0\}$), then $f''(x) = 1/x^2 > 0 \; \forall x$
- **Afine function.** Let $f : \mathbb{R}^n \to \mathbb{R}, f(x) = b^T x + c$ for some $b \in \mathbb{R}^n, c \in \mathbb{R}$. In this case the Hessian, $\nabla_x^2 f(x) = 0 \; \forall x$. Since the zero matrix is both PSD and NSD, f is both convex and concave
- **Quadratic functions.** Let $f : \mathbb{R}^n \to \mathbb{R}, f(x) = \frac{1}{2} x^T A x + b^T x + c$, $A \in \mathbb{S}^n, b \in \mathbb{R}^n$ and $c \in \mathbb{R}$. Since $\nabla_x^2 f(x) = A$, the convexity or non-convexity of f is determined whether A is PSD
  Squared Euclidean norm $f(x) = ||x||_2 = x^T x$ is a special case of quadratic functions where $A = I, b = 0, c = 0$, so it is strictly convex function

Convex Optimization Problems

A convex optimization problem can be formally written as

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \; i = 1, \ldots, m \\ & h_i(x) = 0, \; i = 1, \ldots, p \end{aligned}$$

where f, g_i are convex function, and h_i are affine functions, and x is optimizable variable

- **Locally optimal.** A point x is locally optimal if it is feasible (i.e., it satisfies the constraints of the optimization problem) and if there exists some $R > 0$ such that all feasible points z with $||x - z||_2 \leq R$, satisfy $f(x) \leq f(z)$
- A point x is globally optimal if it is feasible and for all feasible points z, $f(x) \leq f(z)$

Special Cases of Convex Problems

Linear Programming

We say that a convex optimization problem is a **linear program** (LP) if both the objective function f and inequality constraints g_i are affine function.

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^T x + d \\ \text{such that} \quad & Gx \preceq h \\ & Ax = b \end{aligned}$$

where $x \in \mathbb{R}^n$ is the optimization variable, $c \in \mathbb{R}^n, d \in \mathbb{R}, G \in \mathbb{R}^{m \times n}, h \in \mathbb{R}^m, A \in \mathbb{R}^{p \times n}, b \in \mathbb{R}^p$, and $\preceq$ denotes elementwise inequality

Special Cases of Convex Problems

Quadratic Programming

We say that a convex optimization problem is a **quadratic program** (QP) if inequality constraints g_i are all affine, but if the objective function f is convex quadratic function. Formally,

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2}x^T Px + c^T x + d \\ \text{such that} \quad & Gx \preceq h \\ & Ax = b \end{aligned}$$

where $x \in \mathbb{R}^n$ is the optimization variable, $c \in \mathbb{R}^n, d \in \mathbb{R}, G \in \mathbb{R}^{m \times n}, h \in \mathbb{R}^m, A \in \mathbb{R}^{p \times n}, b \in \mathbb{R}^p$, and $P \in \mathbb{S}^n_+$, a symmetric positive semidefinite matrix.

Why to bother about the convexity of a function?

- For a convex optimization problem, we have only one local optimal solution which is also the global optimal solution
- For a non-convex optimization problem, there are many local optimal solutions and it can take a lot of time to identify whether the problem has no solution or if the solution is global
- Efficiency in time of the convex optimization problem is much better
- Convex problem usually is much more easier to deal with in comparison to a non convex problem which takes a lot of time and it might lead you to a dead end.

- One way to represent a classifier is by using discriminant functions
- It works for both binary and multi-class classification
- Idea
  - For every class $i = 0, 1, 2, \ldots, k$, define a function $g_i(\mathbf{x}) : X \mapsto \mathbb{R}$
  - When the decision on input $\mathbf{x}$ should be made choose the class with highest value of $g_i(\mathbf{x})$

Optimal Hyperplane for Linearly Separable Patterns

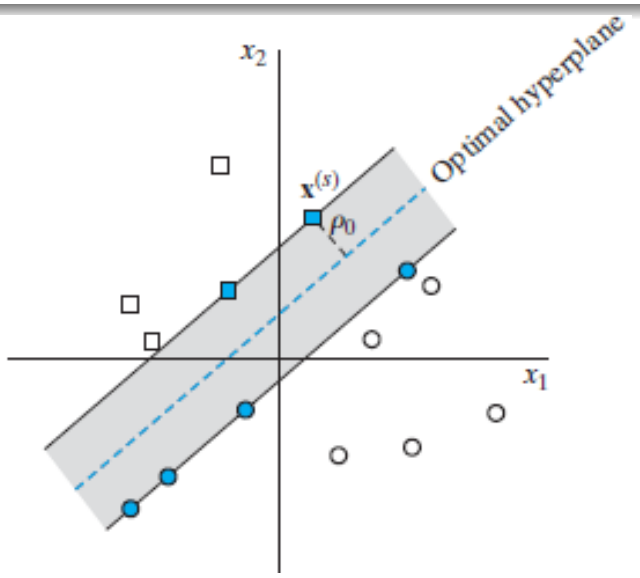
- Consider the training sample $\{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$, and $d_i \in \{+1, -1\}$
- Equation of a decision surface in the form of a hyperplane that does the separation

$$\mathbf{w}^T\mathbf{x} + b = 0 \tag{1}$$

we may write

$$\begin{aligned} \mathbf{w}^T\mathbf{x}_i + b \geq 0 \quad d_i = +1 \\ \mathbf{w}^T\mathbf{x}_i + b < 0 \quad d_i = -1 \end{aligned} \tag{2}$$

- For a given weight vector $\mathbf{w}$ and bias b, the separation between the hyperplane defined in eq(1) and the closest data point is called the margin of separation, ρ
- Goal of a SVM is to find hyperplane for which ρ is maximized
- Under this condition the decision surface is referred to as the optimal hyperplane



- Let $\mathbf{w}_o$ and b_o denote the optimum values of weight vector and bias
- Optimal hyperplane, representing multidimensional linear decision surface in the input space is defined by

$$\mathbf{w}_o^T \mathbf{x} + b_o = 0 \tag{3}$$

- The [1]discriminant function

$$g(\mathbf{x}) = \mathbf{w}_o^T \mathbf{x} + b_o \tag{4}$$

gives an algebraic measure of the distance from $\mathbf{x}$ to the optimal hyperplane

- $\mathbf{x}$ can be expressed as, $\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}_o}{\|\mathbf{w}_o\|}$, where $\mathbf{x}_p$ is the normal projection of $\mathbf{x}$ onto the optimal hyperplane, and r is the desired algebraic distance
- $r = \begin{cases} +ve & \mathbf{x} \text{ on +ve side of plane} \\ -ve & \mathbf{x} \text{ on -ve side of plane} \end{cases}$

[1] It is a statistical analysis to predict a categorical dependent variable by one or more continuous or binary independent variable called predictor variables

- By definition, $g(\mathbf{x}_p) = 0$, it follows that

$$g(\mathbf{x}) = \mathbf{w}_o^T \mathbf{x} + b_o = r\|\mathbf{w}_o\|$$

or $r = \dfrac{g(\mathbf{x})}{\|\mathbf{w}_o\|}$

- Distance from the origin to the optimum hyperplane is given by $b_o/\|\mathbf{w}_o\|$
- $Origin =$

$$\begin{cases} \text{on the positive side of hyperplane} & b_o > 0 \\ \text{on the negative side of hyperplane} & b_o < 0 \\ \text{optimal hyperplane passes through the origin} & b_o = 0 \end{cases}$$

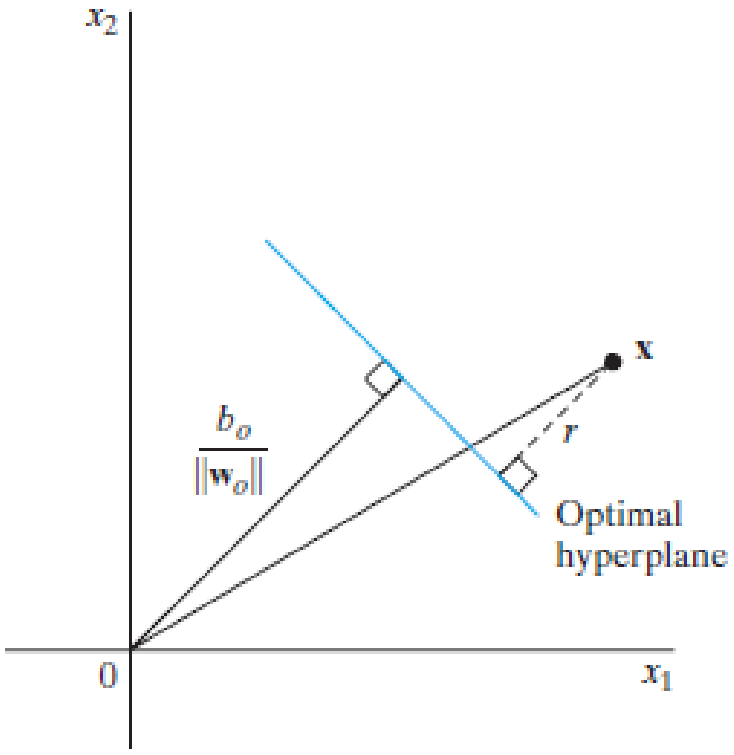
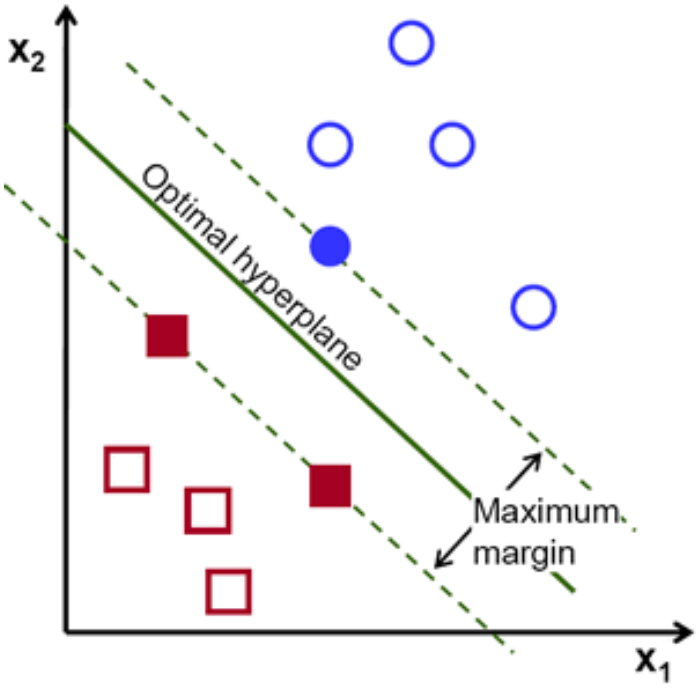
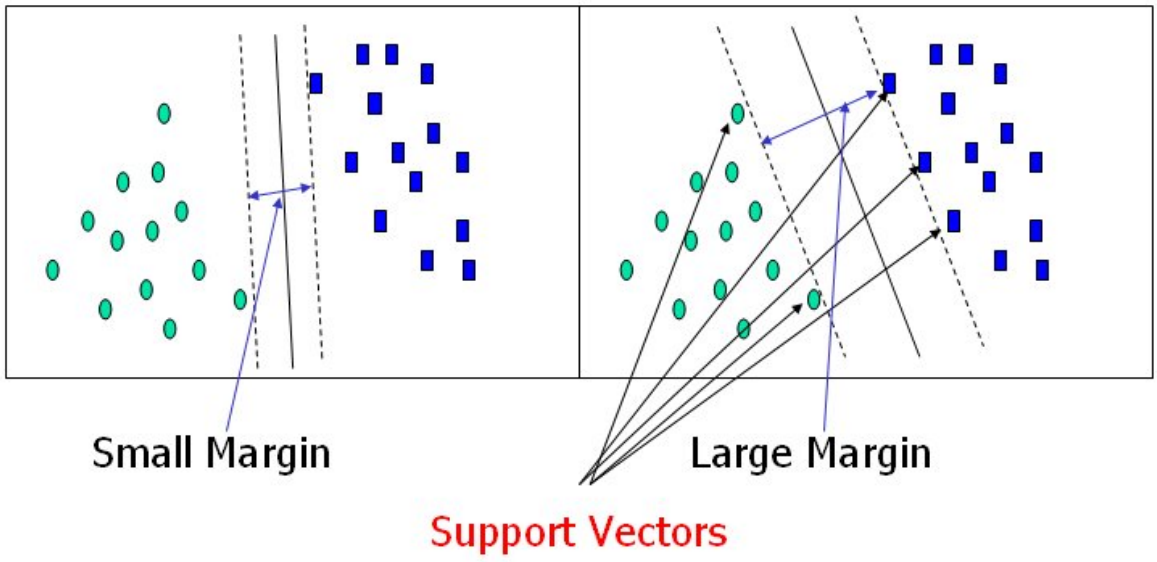


Figure 1: Geometric interpretation of algebraic distance of points to the optimal hyperplane for a two-dimensional case.



- **Support vectors**:input points closest to the separating hyperplane
- **Margin of separation** ρ: distance between the separating hyperplane and the closest input point



- It is required to find $\mathbf{w}_o$ and b_o for the optimal hyperplane, for the given training set $\mathcal{T} = \{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$
- Optimal hyperplane is supposed to maximize the margin of separation ρ
- With that requirement, we can write the conditions that $\mathbf{w}_o$ and b_o must meet:

$$\mathbf{w}_o^T \mathbf{x}_i + b_o \geq +1 \quad d_i = +1$$
$$\mathbf{w}_o^T \mathbf{x}_i + b_o \leq -1 \quad d_i = -1$$

- $\geq +1$ and ≤ -1 and support vectors are those $\mathbf{x}^{(s)}$ where equality holds i.e., $\mathbf{w}_o^T \mathbf{x}^{(s)} + b_o = \pm 1$
- Since $r = (\mathbf{w}_o^T \mathbf{x} + b_o)/\|\mathbf{w}_o\|$,

$$r = \begin{cases} \dfrac{1}{\|\mathbf{w}_o\|} & d_i = +1 \\ \dfrac{-1}{\|\mathbf{w}_o\|} & d_i = -1 \end{cases}$$

- Margin of separation between two classes is

$$\rho = 2r = \frac{2}{\|\mathbf{w}_o\|}$$

- Thus, maximizing the margin of separation between two classes is equivalent to minimizing the Euclidean norm of the weight vector $\mathbf{w}$
- Goal is to develop a computationally efficient procedure for using the training sample $\mathcal{T}$ to find the optimal hyperplane, subject to the constraint

$$d_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \quad i = 1, 2, \ldots, N$$

Quadratic Optimization for Finding Optimal Hyperplane

Problem Statement

Given the training sample $\{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$, find the optimum values of the weight vector $\mathbf{w}$ and bias b such that they satisfy the constraints

$$d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad i = 1, 2, \ldots, N$$

and the weight vector $\mathbf{w}$ minimizes the cost function

$$\Phi(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

This optimization problem is called primal problem and it is characterized as follows:

- Cost function $\Phi(\mathbf{w})$ is a convex function of $\mathbf{w}$
- Constraints are linear in $\mathbf{w}$

- We can solve this constraint optimization problem using the method of Lagrange multipliers
- We write the Lagrange function as

$$J(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N}\alpha_i[d_i(\mathbf{w}^T\mathbf{x}_i + b) - 1]$$

- Conditions of optimality

$$\frac{\partial J(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = \mathbf{0}, \quad \frac{\partial J(\mathbf{w}, b, \alpha)}{\partial b} = 0$$

- Application of optimality condition 1 to Lagrange function yields

$$\mathbf{w} = \sum_{i=1}^{N}\alpha_i d_i \mathbf{x}_i$$

- Application of optimality condition 2 to Lagrange function yields

$$\sum_{i=1}^{N}\alpha_i d_i = 0$$

- At the saddle point, each Lagrange multiplier α_i, the product of that multiplier with its corresponding constraint values vanishes as shown by

$$\alpha_i[d_i(\mathbf{w}^T\mathbf{x}_i + b) - 1] = 0$$

- Thus, only those multiplier exactly meeting above equation can assume nonzero values
- The primal problem deals with convex cost function and linear constraints
- Given such a constrained problem, it is possible to construct another problem called the dual problem
- Dual problem has the same optimal values as the primal problem, but with Lagrange multipliers providing the optimal solution

Duality Theorem

(a) If the primal problem has an optimal solution, the dual problem also has an optimal solution, and the corresponding optimal values are equal

(b) In order for $\mathbf{w}_o$ to be an optimal primal solution and α_o to be an optimal dual solution, it is necessary and sufficient that $\mathbf{w}_o$ is feasible for the primal problem, and

$$\Phi(\mathbf{w}_o) = J(\mathbf{w}_o, b_o, \alpha_o) = \underset{\mathbf{w}}{minimize}\ J(\mathbf{w}, b_o, \alpha_o)$$

To postulate the dual problem from primal problem, we rewrite Lagrange function

- $J(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \alpha_i d_i \mathbf{w}^T\mathbf{x}_i - b\underbrace{\sum_{i=1}^{N} \alpha_i d_i}_{=0} + \sum_{i=1}^{N} \alpha_i$
- Also from $\mathbf{w} = \sum_{i=1}^{N} \alpha_i d_i \mathbf{x}_i$ we can write

$$\mathbf{w}^T\mathbf{w} = \sum_{i=1}^{N} \alpha_i d_i \mathbf{w}^T\mathbf{x}_i = \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T\mathbf{x}_j$$

- Setting the objective function $J(\mathbf{w}, b, \alpha) = Q(\alpha)$, we may write as

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T\mathbf{x}_j$$

Statement of Dual Problem

Given the training sample $\{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$, find the Lagrange multipliers $\{\alpha_i\}_{i=1}^{N}$ that maximize the objective function

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T\mathbf{x}_j$$

subjected to the constraints

(1) $\sum_{i=1}^{N} \alpha_i d_i = 0$

(2) $\alpha_i \geq 0 \quad i = 1, 2, \ldots, N$

- $Q(\alpha)$ to be maximized depends only on the input patterns in the form of a set of dot products $\{\mathbf{x}_i^T \mathbf{x}_j\}_{(i,j)=1}^N$
- Having determined Lagrange multipliers $\alpha_{o,i}$, we can find the optimum weight vector as

$$\mathbf{w}_o = \sum_{i=1}^{N} \alpha_{o,i} d_i \mathbf{x}_i$$

- To compute optimum bias b_o, we make use of $\mathbf{w}_o$ and the relation $\mathbf{w}_o^T \mathbf{x}^{(s)} + b_o = 1$, and thus we get

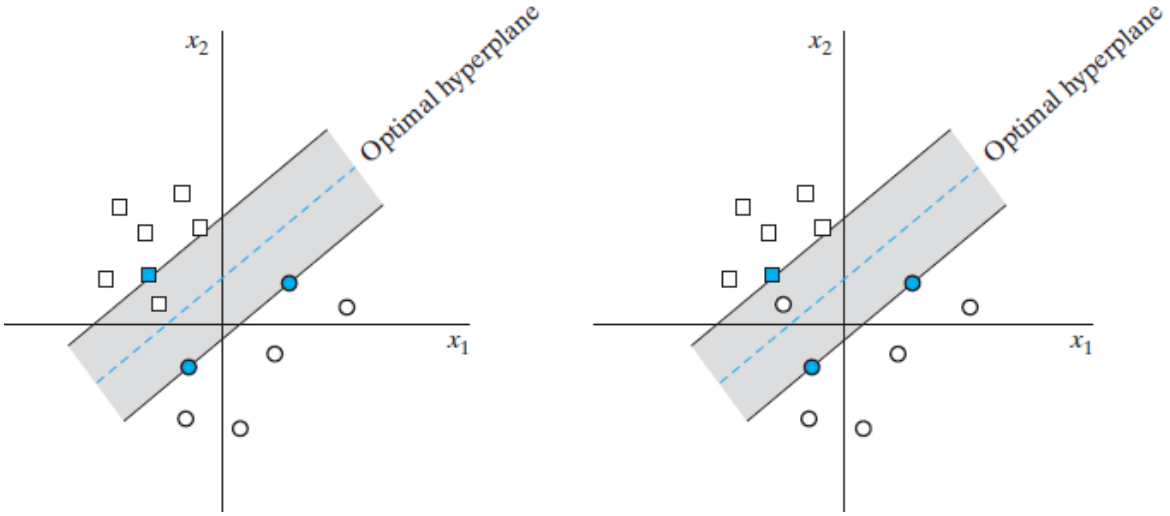
$$b_o = 1 - \mathbf{w}_o^T \mathbf{x}^{(s)}, \quad d^{(s)} = 1$$

Optimal Hyperplane for Nonseparable Patterns

- Lets consider the case of nonseparable patterns
- Given such training data, it is not possible to construct a separating hyperplane without encountering classification error
- We would like to find an optimal hyperplane that minimizes the probability of classification error, averaged over the training data
- Margin of separation between classes is said to be soft if a data point $(\mathbf{x}_i, d_i)$ violates the following condition

$$d_i(\mathbf{w}^T \mathbf{x}_i + b) \geq +1 \qquad i = 1, 2, \ldots, N$$

- These violation can arise in one of two ways:
  - Data point $(\mathbf{x}_i, d_i)$ falls inside the region of separation but on the right side of decision surface
  - Data point $(\mathbf{x}_i, d_i)$ falls on the wrong side of the decision surface



- We have correct classification in case 1, but misclassification in case 2
- Lets modify our definition of separating hyperplane

$$d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \qquad i = 1, 2, \ldots, N$$

- The ξ_i are called slack variables; they measure the deviation of data point from the ideal condition of pattern separability

- For $0 \leq \xi_i \leq 1$, the data point falls inside the region of separation but on the right side of decision surface
- For $\xi > 1$, it falls on the wrong side of the separating hyperplane
- The support vectors are those data points that satisfy

$$d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \qquad i = 1, 2, \ldots, N$$

even if $\xi_i > 0$

- If an example with $\xi_i > 0$ is left out of the training set, the decision surface would change
- Support vectors are defined in exactly the same way for both linearly separable and nonseparable

- Our goal is to find separating hyperplane for which misclassification error averaged on the training set is minimized
- We may do this by minimizing the [2]functional

$$\Phi(\xi) = \sum_{i=1}^{N} I(\xi_i - 1)$$

  wrt $\mathbf{w}$, subjected to separating surface condition and constraint on $\|\mathbf{w}\|^2$
- The function $I(\xi)$ is an indicator function, defined by

$$I(\xi) = \begin{cases} 0 & \xi \leq 0 \\ 1 & \xi > 0 \end{cases}$$

- Minimization of $\Phi(\xi)$ wrt $\mathbf{w}$ is a non-convex optimization problem that is NP-complete

[2]In mathematics, and particularly in functional analysis, a functional is a function from a vector space into its underlying field of scalars

- We simplify the computation by formulating the functional to be minimized wrt $\mathbf{w}$ as follows

$$\Phi(\mathbf{w}, \xi) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \underbrace{C\sum_{i=1}^{N} \xi_i}_{\text{controls errors-upper bound on \# test errors}}$$

- Parameter C controls the tradeoff between complexity of machine and the number of non-separating points
- The functional $\Phi(\mathbf{w}, \xi)$ is optimized wrt $\mathbf{w}$ and $\{\xi_i\}_{i=1}^N$ subjected to the constraint of separating hyperplane

Given the training sample $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$, find the optimum values of the weight vector $\mathbf{w}$ and bias b such that they satisfy the constraint

$$d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad i = 1, 2, \ldots, N$$

$$\xi_i \geq 0 \quad \forall i$$

and such that the weight vector $\mathbf{w}$ and slack variables ξ_i minimizes the cost functional

$$\Phi(\mathbf{w}, \xi) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N} \xi_i$$

where C is a user-specified parameter.

- Formally, we can represent this optimization problem as

$$\begin{aligned} \underset{\mathbf{w},\xi}{\text{minimize}} \quad & \Phi(\mathbf{w},\xi) \\ \text{subject to} \quad & d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad i = 1,2,\ldots,N \\ & \xi_i \geq 0 \quad \forall i \end{aligned}$$

- Using method of Lagrange multiplier and proceeding in the manner followed for the separable case, we may formulate the dual problem for nonseparable patterns as

Given the training set $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$, find the Lagrange multipliers $\{\alpha_i\}_{i=1}^N$ that maximize the objective function

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j d_i d_j \mathbf{x}_i^T\mathbf{x}_j$$

subjected to the constraints

(1) $\sum_{i=1}^{N} \alpha_i d_i = 0$

(2) $0 \leq \alpha_i \leq C \quad i = 1,2,\ldots,N$

where C is a user-specified positive number.

- Note that neither the slack variables ξ_i nor their Lagrange multipliers appear in the dual problem
- Dual problem for the case of nonseparable is similar to that of the linearly separable case
- The optimum solution for the weight vector w is given by

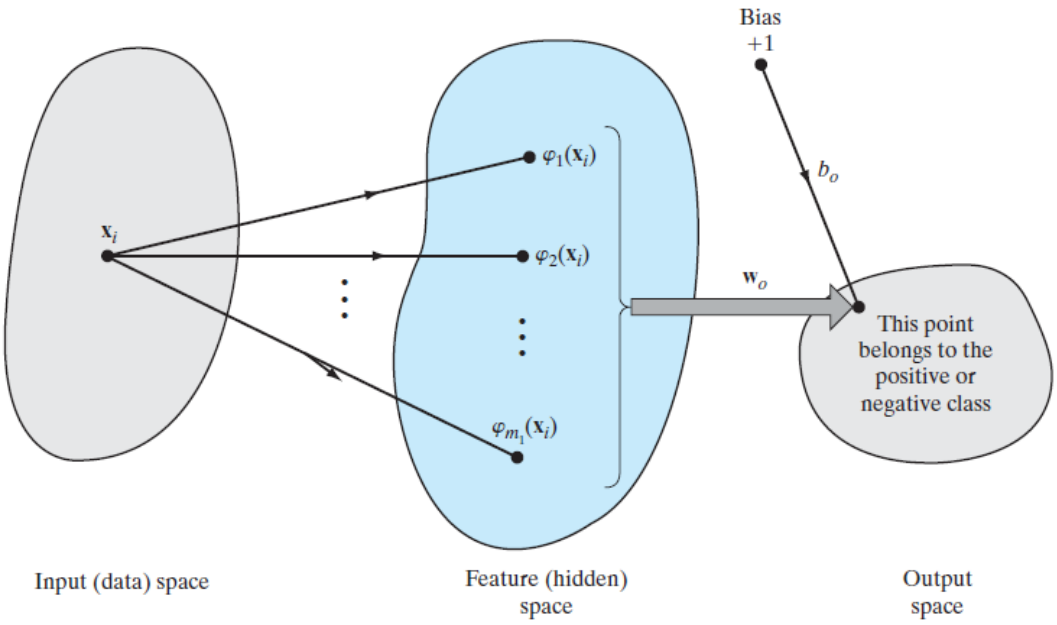
$$\mathbf{w}_o = \sum_{i=1}^{N_s} \alpha_{o,i} d_i \mathbf{x}_i$$

,

$$b_o = d_i^{-1}(1 - \xi_i) - \mathbf{w}_o^T\mathbf{x}_i$$

How to Build a SVM for Pattern Recognition

- We know how to find the optimal hyperplane for nonseparating patterns
- The idea of SVM hinges on two mathematical operations for pattern-recognition task
  1. nonlinear mapping of input vector into a high-dimensional feature space that is hidden from both the input and output
  2. construction of an optimal hyperplane for separating the features discovered in step 1
  3. number of features constituting the hidden space is determined by the number of support vectors
  4. SVM theory provides an analytic approach for determining the optimum size of the feature space



SVM Viewed as Kernel Machine

- Let $\mathbf{x} \in X \wedge \mathbf{x} \in \mathbb{R}^{m_0}$
- $\{\varphi_j(\mathbf{x})\}_{j=1}^{m_1}$: set of nonlinear transformation from input space to the feature space
- Assumption: $\varphi_j(\mathbf{x})$ is defined apriori for all j
- Given such a set of transformations, we can define a hyperplane acting as the decision surface as follows

$$\sum_{j=1}^{m_1} w_j \varphi_j(\mathbf{x}) + b = 0$$

  where $\{w_j\}_{j=1}^{m_1}$ denotes a set of linear weights connecting the feature space to the output space, and b bias
- Also, we can write above equation as

$$\sum_{j=0}^{m_1} w_j \varphi_j(\mathbf{x}) = 0$$

  . It defines the decision surface computed in the feature space

- Define vector $\varphi(\mathbf{x}) = [\varphi_0(\mathbf{x}), \varphi_1(\mathbf{x}), \ldots, \varphi_{m_1}(\mathbf{x})]^T$ and by definition $\varphi_0(\mathbf{x}) = 1 \quad \forall \mathbf{x}$
- The vector $\varphi(\mathbf{x})$ represents the "image" induced in the feature space due to the input vector $\mathbf{x}$
- Thus, in terms of this image, we may define the decision surface in the compact form:

$$\mathbf{w}^T \varphi(\mathbf{x}) = 0$$

- Now, adapting equation $\mathbf{w} = \sum_{i=1}^{N} \alpha_i d_i \mathbf{x}_i$ to our situation involving a feature space where we 'seek' linear separability of features, we may write

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i d_i \varphi(\mathbf{x}_i)$$

- So we may define the decision surface computed in the feature space as

$$\sum_{i=1}^{N} \alpha_i d_i \varphi^T(\mathbf{x}_i)\varphi(\mathbf{x}) = 0$$

- The term $\varphi^T(\mathbf{x}_i)\varphi(\mathbf{x})$ represents the inner product of two vectors induced in the feature space by $\mathbf{x}$ and $\mathbf{x}_i$
- Define a inner-product kernel, denoted by $\mathbf{K}(\mathbf{x}, \mathbf{x}_i)$ as

$$\mathbf{K}(\mathbf{x}, \mathbf{x}_i) = \varphi^T(\mathbf{x})\varphi(\mathbf{x}_i) = \sum_{j=0}^{m_1} \varphi_j(\mathbf{x})\varphi_j(\mathbf{x}_i) \quad i = 1, 2, \ldots, N$$

- Kernel is formally defined as

Definition

The kernel $\mathbf{K}(\mathbf{x}, \mathbf{x}_i)$ is a function that computes the inner product of the images produced in the feature space under the embedding ϕ of two data points in the input space.

Kernel $K(x, x_i)$ has following properties

1. The function $k(x, x_i)$ is symmetric about the center point x_i, that is

$$k(x, x_i) = k(x_i, x) \quad \forall x_i$$

   and it attains its maximum value at the point $x = x_i$.
2. The total volume under the surface of the function $K(x, x_i)$ is a constant.

The Kernel Trick

- As far as pattern classification in output space is concerned, specifying the kernel $k(x, x_i)$ is sufficient; we don't need to explicitly compute the weight vector w_o. Due to this reason application of

$$k(x, x_i) = \sum_{j=1}^{N} \varphi_j(x_i)\varphi_j(x)$$

  is commonly referred to as the kernel trick
- Even though we assumed that the feature space could be of infinite dimension, the linear equation

$$\sum_{i=1}^{N} \alpha_i d_i k(x, x_i) = 0$$

  defining the optimal hyperplane, consists of a finite number of terms that is equal to the number of training patterns used in the classifier

- SVM when designed with kernel is known as a kernel machine
- For pattern classification, the machine is parameterized by an N-dimensional vector whose ith term is defined by the product

$$\alpha_i d_i, \quad i = 1, 2, \ldots, N$$

- $k(x_i, x_j)$ can be viewed as the ij-th element of the symmetric $N \times N$ matrix

$$K = \{k(x_i, x_j)\}_{i,j=1}^{N}$$

  where K is a nonnegative definitive matrix called the kernel matrix
- K is nonnegative definitive or positive semidefinite in that it satisfies the condition

$$x^T K x \geq 0$$

  for any real-valued vector x whose dimension is compatible with K

Mercer's Theorem

Theorem

Let $k(x, x')$ be a continuous symmetric kernel that is defined in the closed interval $a \leq x \leq b$, and likewise for x'. The kernel $k(x, x')$ can be expanded in the series

$$k(x, x') = \sum_{j=1}^{\infty} \lambda_i \varphi_i(x) \varphi_i(x')$$

with positive coefficients $\lambda_i > 0$ for all i. For this expansion to be valid and for it to converge absolutely and uniformly, it is necessary and sufficient that the condition

$$\int_b^a \int_b^a k(x, x') \psi(x) \psi(x') dx dx' \geq 0$$

holds for all $\psi(.)$, for which we have

$$\int_b^a \psi^2(x) dx < \infty.$$

- The features $\varphi_i(x)$ are called [3]eigenfunctions of the expansion, and λ_i are called eigenvalues
- The fact that all the eigenvalues are positive means that the kernel $k(x, x')$ is positive definite
- Mercer's theorem tells us whether a candidate kernel is actually an inner-product kernel in some space, and therefore admissible for use in SVM.
- Mercer's theorem says nothing about how to construct the functions $\varphi_i(x)$; we have to do ourselves

[3] An eigenfunction is an eigenvector that is also a function. Thus, an eigenfunction is an eigenvector but an eigenvector is not necessarily an eigenfunction. See https://goo.gl/vZxn2I

Design of Support Vector Machines

- The expansion of kernel $k(x, x_i) = \sum_{j=1}^{\infty} \varphi_j(x_i)\varphi_j(x)$ permits us to construct a decision surface that is nonlinear in the input space, but whose image in the feature space is linear
- Now we may state the dual form for the constrained optimization of a SVM as follows

Given the training sample $\{(x_i, d_i)\}_{i=1}^{N}$, find the Lagrange multipliers $\{\alpha_i\}_{i=1}^{N}$ that maximize the objective function

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j k(x_i, x_j)$$

subject to the constraints

- $\sum_{i=1}^{N} \alpha_i d_i = 0$
- $0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, N$ where C is a user-specified positive constants

Examples of SVM

Table 1: Summary of Inner-Product Kernels

Type of SVM	$k(x, x_i), \ i = 1, 2, \ldots, N$	Comments
Polynomial learning machine	$(x^T x_i + 1)^p$	Power p is specified a priori by the user
Radial-basis-function network	$\exp\left(-\frac{1}{2\sigma^2}\|x - x_i\|^2\right)$	Width σ^2 common to all the kernels, is specified apriori by the user
Two-layer perceptron	$\tanh(\beta_0 x^T x_i + \beta_1)$	Mercer's theorem is satisfied only for some values of β_0 and β_1

- Inner-product kernel for polynomial and RBF types of SVM satisfy Mercer's theorem
- For all three machine types, the dimensionality of the feature space is determined by the number of support vectors extracted from the training data by the solution of constrained optimization problem
- SVM avoids the need for heuristics often used in the design of RBFN and MLP
- In RBF type SVM, the number of radial basis functions and their centers are determined automatically by the number of support vectors and their values

SVM offers a solution to the design of a learning machine by controlling model complexity independent of dimensionality

- **Conceptual problem** Dimensionality of the feature space is purposely made very large to enable the construction of a decision surface in the form of a hyperplane in that space

  For good generalization performance, the model complexity is controlled by imposing certain constraints on the construction of the separating soft-margin hyperplane, which results in the extraction of a fraction of the training data as support vectors
- **Computational problem** the need to compute the weight vector and the bias in the output layer of the RBF network is avoided by using the kernel trick

XOR Problem

Table 2: XOR Problem

Input vetor x	Desired response d
(-1,-1)	-1
(-1,+1)	+1
(+1,-1)	+1
(+1,+1)	-1

- Let $k(x, x_i) = (1 + x^T x_i)^2$
- With $x = [x_1, x_2]^T$ and $x_i = [x_{i1}, x_{i2}]^T$, we can write

$$x^T x_i = x_1 x_{i1} + x_2 x_{i2}$$

and

$$k(x, x_i) = 1 + x_1^2 x_{i1}^2 + 2x_1 x_2 x_{i1} x_{i2} + x_2^2 x_{i2}^2 + 2x_1 x_{i1} + 2x_2 x_{i2}$$

- We know that $k(x, x_i) = \varphi(x)^T \varphi(x_i)$
- Image of input vector x induced in the feature space is deduced to be

$$\varphi(x) = [1, x_1^2, \sqrt{2}x_1x_2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2]^T$$

similarly

$$\varphi(x_i) = [1, x_{i1}^2, \sqrt{2}x_{i1}x_{i2}, x_{i2}^2, \sqrt{2}x_{i1}, \sqrt{2}x_{i2}]^T, \quad i = 1, 2, 3, 4$$

- $K = \{k(x_i, x_j)\}_{i,j=1}^N$, here $N = 4$ so

$$K = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \\ k_{41} & k_{42} & k_{43} & k_{44} \end{bmatrix}$$

- Equation
$k(x, x_i) = 1 + x_1^2 x_{i1}^2 + 2x_1x_2x_{i1}x_{i2} + x_2^2 x_{i2}^2 + 2x_1x_{i1} + 2x_2x_{i2}$ can be written as

$$k(x_i, x_j) = 1 + x_{i1}^2 x_{j1}^2 + 2x_{i1}x_{i2}x_{j1}x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$$

to find the ij-th element of the kernel matrix K. Using this equation, we get the following kernel matrix

$$K = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}$$

- Objective function for the dual problem is given by

$$Q(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j k(x_i, x_j)$$

Expanding

$$Q(\alpha) = \sum_{i=1}^{4} \alpha_i - \frac{1}{2} \sum_{i=1}^{4} \sum_{j=1}^{4} \alpha_i \alpha_j d_i d_j k(x_i, x_j)$$

, we get

$$\begin{aligned} Q(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}[&\alpha_1\alpha_1 d_1 d_1 k_{11} + \alpha_1\alpha_2 d_1 d_2 k_{12} \\ &+ \alpha_1\alpha_3 d_1 d_3 k_{13} + \alpha_1\alpha_4 d_1 d_4 k_{14} + \alpha_2\alpha_1 d_2 d_1 k_{21} \\ &+ \alpha_2\alpha_2 d_2 d_2 k_{22} + \alpha_2\alpha_3 d_2 d_3 k_{23} + \alpha_2\alpha_4 d_2 d_4 k_{24} + \alpha_3\alpha_1 d_3 d_1 k_{31} \\ &+ \alpha_3\alpha_2 d_3 d_2 k_{32} + \alpha_3\alpha_3 d_3 d_3 k_{33} + \alpha_3\alpha_4 d_3 d_4 k_{34} + \alpha_4\alpha_1 d_4 d_1 k_{41} \\ &+ \alpha_4\alpha_2 d_4 d_2 k_{42} + \alpha_4\alpha_3 d_4 d_3 k_{43} + \alpha_4\alpha_4 d_4 d_4 k_{44}] \end{aligned}$$

$$\begin{aligned} Q(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 \\ - \frac{1}{2}[&\alpha_1^2 d_1^2 k_{11} + \alpha_1\alpha_2 d_1 d_2 k_{12} + \alpha_1\alpha_3 d_1 d_3 k_{13} + \alpha_1\alpha_4 d_1 d_4 k_{14} \\ &+ \alpha_2\alpha_1 d_2 d_1 k_{21} + \alpha_2^2 d_2^2 k_{22} + \alpha_2\alpha_3 d_2 d_3 k_{23} + \alpha_2\alpha_4 d_2 d_4 k_{24} \\ &+ \alpha_3\alpha_1 d_3 d_1 k_{31} + \alpha_3\alpha_2 d_3 d_2 k_{32} + \alpha_3^2 d_3^2 k_{33} + \alpha_3\alpha_4 d_3 d_4 k_{34} \\ &+ \alpha_4\alpha_1 d_4 d_1 k_{41} + \alpha_4\alpha_2 d_4 d_2 k_{42} + \alpha_4\alpha_3 d_4 d_3 k_{43} + \alpha_4^2 d_4^2 k_{44}] \end{aligned}$$

Substituting for k_{ij} from the kernel matrix K in

$$\begin{aligned} Q(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 \\ - \frac{1}{2}[&\alpha_1^2 d_1^2 k_{11} + \alpha_1\alpha_2 d_1 d_2 k_{12} + \alpha_1\alpha_3 d_1 d_3 k_{13} + \alpha_1\alpha_4 d_1 d_4 k_{14} \\ &+ \alpha_2\alpha_1 d_2 d_1 k_{21} + \alpha_2^2 d_2^2 k_{22} + \alpha_2\alpha_3 d_2 d_3 k_{23} + \alpha_2\alpha_4 d_2 d_4 k_{24} \\ &+ \alpha_3\alpha_1 d_3 d_1 k_{31} + \alpha_3\alpha_2 d_3 d_2 k_{32} + \alpha_3^2 d_3^2 k_{33} + \alpha_3\alpha_4 d_3 d_4 k_{34} \\ &+ \alpha_4\alpha_1 d_4 d_1 k_{41} + \alpha_4\alpha_2 d_4 d_2 k_{42} + \alpha_4\alpha_3 d_4 d_3 k_{43} + \alpha_4^2 d_4^2 k_{44}] \end{aligned}$$

we get

$$\begin{aligned} Q(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}[&9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha_1\alpha_4 \\ &+ 9\alpha_2^2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2] \end{aligned}$$

Optimizing $Q(\alpha)$ wrt α_i (i.e $\partial Q(\alpha)/\partial \alpha_i = 0$) yields the following set of simultaneous equations:

$$\begin{aligned} 9\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 &= 1 \\ -\alpha_1 + 9\alpha_2 + \alpha_3 - \alpha_4 &= 1 \\ -\alpha_1 + \alpha_2 + 9\alpha_3 - \alpha_4 &= 1 \\ \alpha_1 - \alpha_2 - \alpha_3 + 9\alpha_4 &= 1 \end{aligned}$$

which can be written in matrix form as

$$\begin{bmatrix} 9 & -1 & -1 & 1 \\ -1 & 9 & 1 & -1 \\ -1 & 1 & 9 & -1 \\ 1 & -1 & -1 & 9 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Hence, the optimum values of the Lagrange multipliers are

$$\alpha_{o,1} = \alpha_{o,2} = \alpha_{o,3} = \alpha_{o,4} = \frac{1}{8}$$

This indicates that in this example, all input vectors are support vectors

The optimum value of $Q(\alpha)$ is

$$Q_o(\alpha) = \frac{1}{4}$$

Correspondingly, we may write

$$\frac{1}{2}\|w_o\|^2 = \frac{1}{4}$$

or

$$\|w_o\| = \frac{1}{\sqrt{2}}$$

Since, $w_o = \sum_{i=1}^{N} \alpha_{o,i} d_i \varphi(x_i)$, we find that the optimum weight vector is

$$w_o = \frac{1}{8}[-\varphi(x_1) + \varphi(x_2) + \varphi(x_3) - \varphi(x_4)]$$

$$= \frac{1}{8}\left[-\begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ -\sqrt{2} \\ -\sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ -\sqrt{2} \\ \sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ \sqrt{2} \\ -\sqrt{2} \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ \sqrt{2} \\ \sqrt{2} \end{bmatrix} \right]$$

$$= \begin{bmatrix} 0 \\ 0 \\ -1/\sqrt{2} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The first element of w_o indicates that the bias b is zero. The optimal hyperplane is defined by

$$w_o^T \varphi(x) = 0$$

Thus

$$\left[0, 0, \frac{-1}{\sqrt{2}}, 0, 0, 0\right] \begin{bmatrix} 1 \\ x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \end{bmatrix} = 0$$

which reduces to

$$-x_1x_2 = 0$$

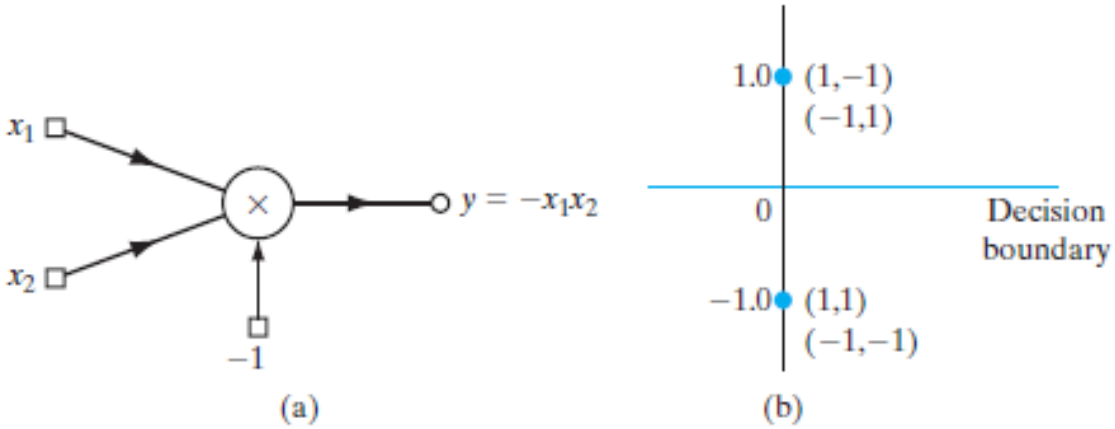


Figure 2: (a) Polynomial machine for solving XOR problem. (b) Induced images in the feature space due to four data points of the XOR problem