Support-Vector Networks

CORINNA CORTES

VLADIMIR VAPNIK

Abstract

Overview

Support-vector network followed the important idea:

1. Input vectors are non-linearly mapped to a very high dimension feature space.

2. A linear decision surface is constructed in the feature space.

3. Special properties of the decision surface ensures high generalization ability of the learning machine.

Introduction

Several Traditional Classification Methods

Fisher's linear discriminant function (1936)

1. Limitation of Linear Assumption: Fisher's method assumes that data is linearly separable. However, in practical applications, data is often non-linearly distributed, leading to poor classification performance on complex datasets.

2. Excessive Number of Parameters: When the dataset is small, the estimation of the covariance matrix may be unreliable, resulting in reduced classification accuracy.

Rosenblatt's Perceptron (1962)

Fixed Hidden Layer Weights: The training method of the perceptron in **Rosenblatt's time** only adjusts the weights of the output layer, while the weights of the hidden layer are fixed. This limitation prevents the perceptron from learning complex non-linear patterns.

Backpropagation Algorithm (1986)

Introduction

Two problems arise

One Conceptual Problem

The conceptual problem is how to find a separating hyperplane that will generalize well: the dimensionality of the feature space will be huge, and not all hyperplanes that separate the training data will necessarily generalize well.

One Technical Problem

The technical problem is how computationally to treat such high-dimensional spaces: to construct polynomial of degree 4 or 5 in a 200 dimensionai space it may be necessary to construct hyperplanes in a billion dimensional feature space.

Introduction

Solutions to Two problems

One Conceptual Problem

The conceptual part of this problem was solved in 1965 (Vapnik, 1982) for the case of *optimal hyperplanes* for separable classes.

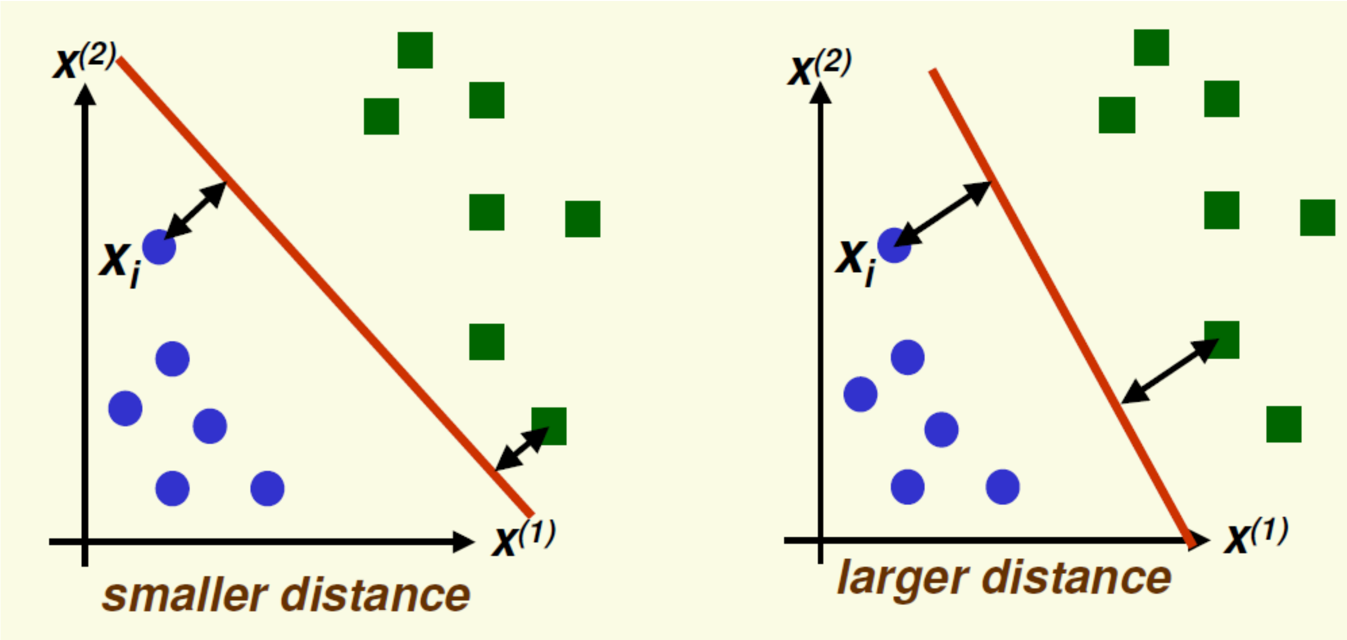
One Technical Problem

In 1992 it was shown (Boser, Guyon, & Vapnik, 1992), that *the order of operations for constructing a decision function can be interchanged*: instead of making a non-linear transformation of the input vectors followed by dot-products with support vectors in feature space, one can first compare two vectors in input space, and then make a non-linear transformation of the value of the result.

Support Vector Machines

SVM

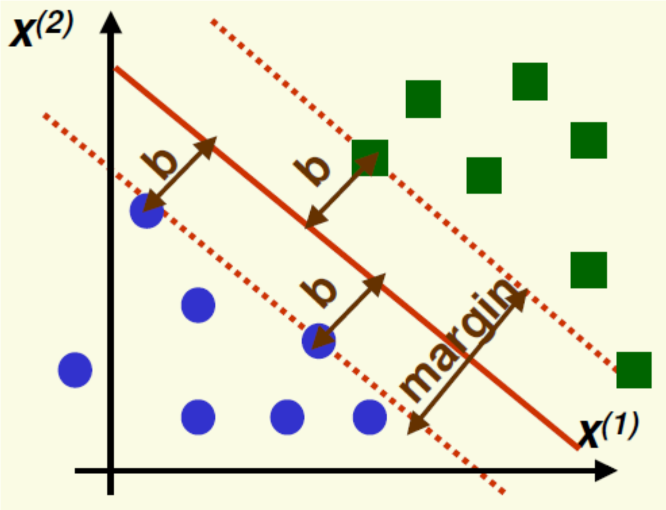
- Idea: maximize distance to the ***closest*** example



- For the optimal hyperplane
  - distance to the closest negative example = distance to the closest positive example

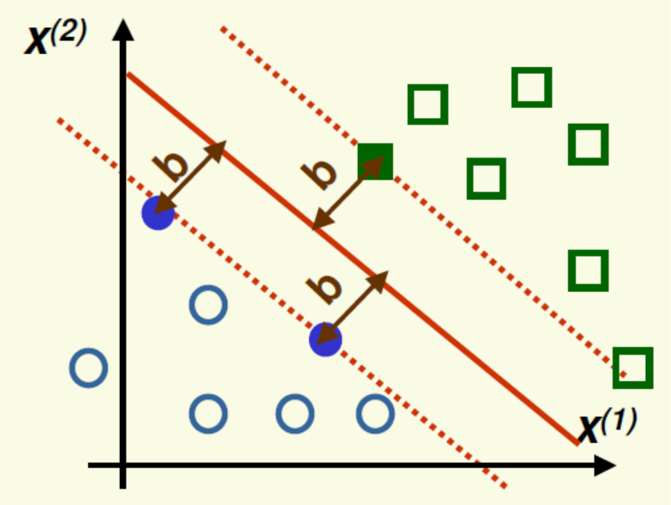
SVM: Linearly Separable Case

- SVM: maximize the margin



- The *margin* is twice the absolute value of distance ***b*** of the closest example to the separating hyperplane
- Better generalization (performance on test data)
  - in practice
  - and in theory

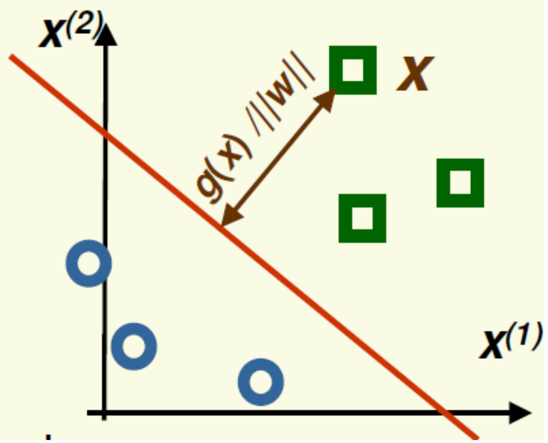
SVM: Linearly Separable Case



- ***Support vectors*** are the samples closest to the separating hyperplane
  - They are the most difficult patterns to classify
  - Recall perceptron update rule
- Optimal hyperplane is completely defined by support vectors
  - Of course, we do not know which samples are support vectors without finding the optimal hyperplane

SVM: Formula for the Margin

$$g(x) = w^t x + w_0$$



- Absolute distance between **x** and the boundary **g(x) = 0**

$$\frac{|w^t x + w_0|}{\|w\|}$$

- Distance is unchanged for hyperplane

$$g_1(x) = \alpha g(x) \qquad \frac{|\alpha w^t x + \alpha w_0|}{\|\alpha w\|} = \frac{|w^t x + w_0|}{\|w\|}$$

- Let $\mathbf{x_i}$ be an example closest to the boundary (on the positive side). Set:

$$|w^t x_i + w_0| = 1$$

- Now the largest margin hyperplane is unique

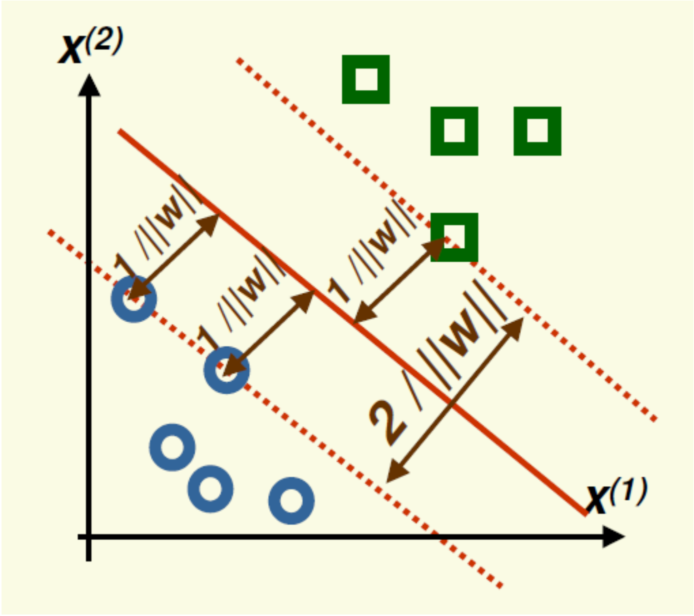
SVM: Formula for the Margin

- For uniqueness, set $|\mathbf{w}^T\mathbf{x}_i+\mathbf{w}_0|=1$ for any sample $\mathbf{x}_i$ closest to the boundary
- The distance from closest sample $\mathbf{x}_i$ to $\mathbf{g(x) = 0}$ is

$$\frac{\left|w^t x_i + w_0\right|}{\|w\|} = \frac{1}{\|w\|}$$

- Thus the margin is

$$\boxed{m = \frac{2}{\|w\|}}$$



SVM: Optimal Hyperplane

- Maximize margin $\boxed{m = \frac{2}{\|w\|}}$
- Subject to constraints $\begin{cases} w^t x_i + w_0 \geq 1 & \text{if } x_i \text{ is positive example} \\ w^t x_i + w_0 \leq -1 & \text{if } x_i \text{ is negative example} \end{cases}$
- Let $\begin{cases} z_i = 1 & \text{if } x_i \text{ is positive example} \\ z_i = -1 & \text{if } x_i \text{ is negative example} \end{cases}$
- Can convert our problem to minimize

$$\boxed{\begin{array}{ll} \text{minimize} & J(w) = \frac{1}{2}\|w\|^2 \\ \text{constrained to} & z_i\left(w^t x_i + w_0\right) \geq 1 \quad \forall i \end{array}}$$

- **J(w)** is a quadratic function, thus there is a single global minimum

SVM: Optimal Hyperplane

- Use Kuhn-Tucker theorem to convert our problem to:
  - Also know as the Karush–Kuhn–Tucker theorem, i.e., the KKT theorem

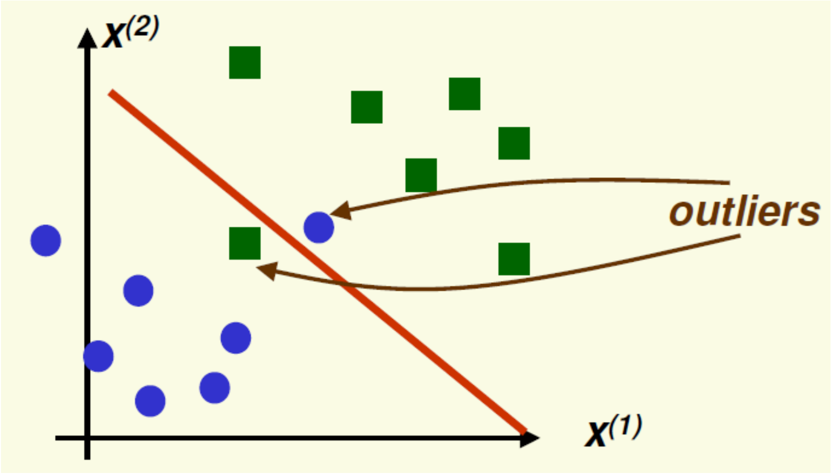
$$\text{maximize} \quad L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j x_i^t x_j$$

$$\text{constrained to} \quad \alpha_i \geq 0 \quad \forall i \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i z_i = 0$$

- ***a*** $=\{a_1, \ldots, a_n\}$ are new variables, one for each sample
- Optimized by quadratic programming

SVM: Non-Separable Case

- Data are most likely to be not linearly separable, but linear classifier may still be appropriate



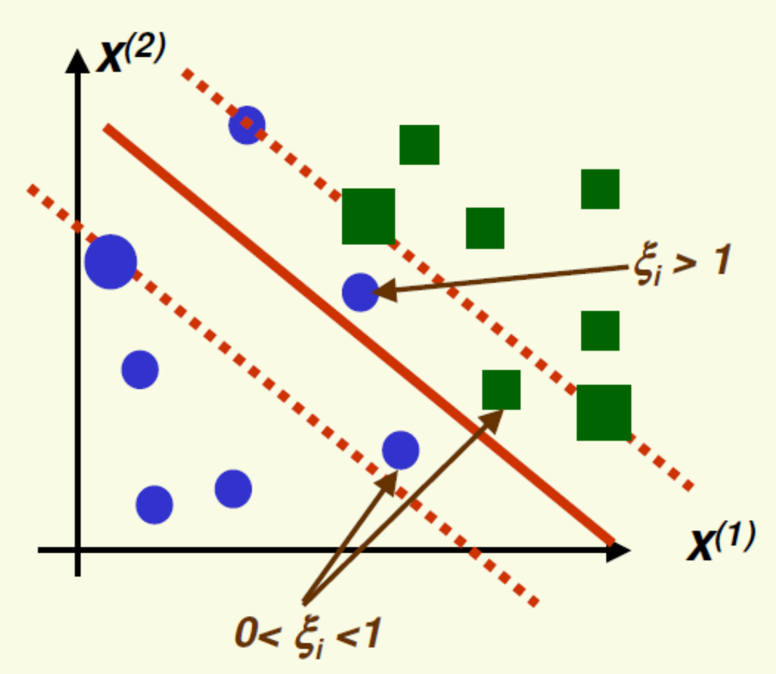
- Can apply SVM in non linearly separable case
- Data should be “almost” linearly separable for good performance

SVM: Non-Separable Case

- Use slack variables $\xi_1,\ldots,\xi_n$ (one for each sample)
- Change constraints from $z_i(w^t x_i + w_0) \geq 1 \quad \forall i$ to

  $z_i(w^t x_i + w_0) \geq 1 - \xi_i \quad \forall i$

- ξ_i is a measure of deviation from the ideal for x_i
  - $\xi_i > 1$: x_i is on the wrong side of the separating hyperplane
  - $0 < \xi_i < 1$: x_i is on the right side of separating hyperplane but within the region of maximum margin
  - $\xi_i < 0$: is the ideal case for x_i



SVM: Non-Separable Case

- We would like to minimize

$$J(w, \xi_1, \ldots, \xi_n) = \frac{1}{2}\|w\|^2 + \beta \sum_{i=1}^{n} I(\xi_i > 0)$$

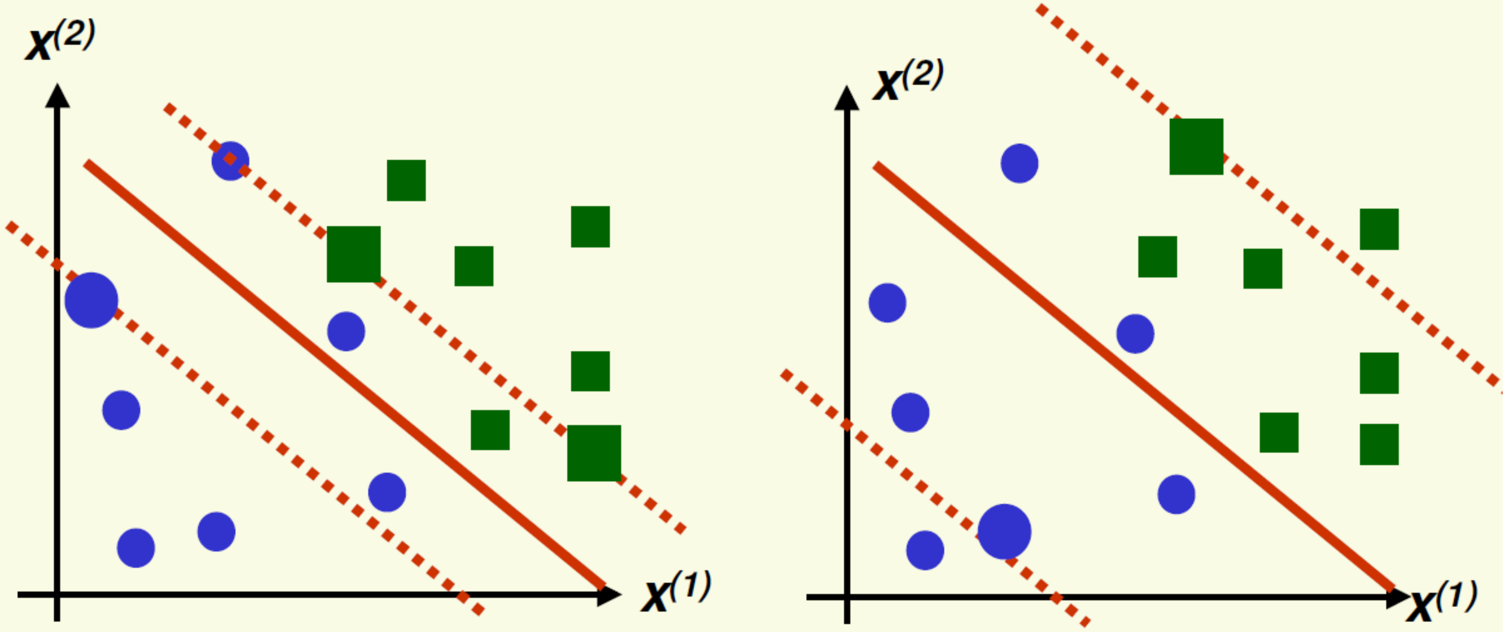
# of samples not in ideal location

- where $I(\xi_i > 0) = \begin{cases} 1 & \text{if } \xi_i > 0 \\ 0 & \text{if } \xi_i \le 0 \end{cases}$
- Constrained to $z_i\left(w^t x_i + w_0\right) \ge 1 - \xi_i$ and $\xi_i \ge 0 \quad \forall i$
- β is a constant that measures the relative weight of first and second term
  - If β is small, we allow a lot of samples to be in not ideal positions
  - If β is large, few samples can be in non-ideal positions

SVM: Non-Separable Case

$$J(w, \xi_1, \ldots, \xi_n) = \frac{1}{2}\|w\|^2 + \beta \sum_{i=1}^{n} I(\xi_i > 0)$$

# of examples not in ideal location



large β, few samples not in ideal position

small β, a lot of samples not in ideal position

SVM: Non-Separable Case

- Unfortunately this minimization problem is NP-hard due to the discontinuity of $I(\xi_i)$
- Instead, we minimize

$$J(w, \xi_1, \ldots, \xi_n) = \frac{1}{2}\|w\|^2 + \beta \sum_{i=1}^{n} \xi_i$$

a measure of # of misclassified examples

- Subject to

$$\begin{cases} z_i\left(w^t x_i + w_0\right) \geq 1 - \xi_i & \forall i \\ \xi_i \geq 0 & \forall i \end{cases}$$

SVM: Non-Separable Case

- Use Kuhn-Tucker theorem to convert to:

$$\text{maximize} \quad L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j x_i^t x_j$$

$$\text{constrained to} \quad 0 \le \alpha_i \le \beta \quad \forall i \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i z_i = 0$$

- **w** is computed using:

$$w = \sum_{i=1}^{n} \alpha_i z_i x_i$$

- Remember that

$$g(x) = \left(\sum_{x_i \in S} \alpha_i z_i x_i \right)^t x + w_0$$

Kernels

- SVM optimization:

maximize
$$L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j x_i^t x_j$$

- Note this optimization depends on samples $\boldsymbol{x_i}$ only through the dot product $\boldsymbol{x_i^t x_j}$
- If we lift $\boldsymbol{x_i}$ to high dimension using $\boldsymbol{\varphi(x)}$, we need to compute high dimensional product $\boldsymbol{\varphi(x_i)^t \varphi(x_j)}$

maximize
$$L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j \underbrace{\varphi(x_i)^t \varphi(x_j)}_{K(x_i, x_j)}$$

- Idea: find kernel function $\mathbf{K(x_i, x_j)}$ s.t. $K(x_i, x_j) = \varphi(x_i)^t \varphi(x_j)$

Choice of Kernel

- Some common choices:
  - Polynomial kernel

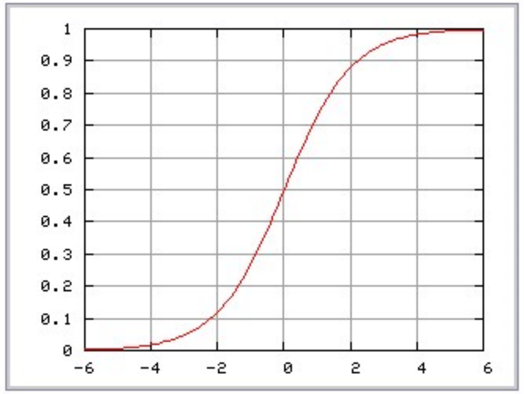
$$K(x_i, x_j) = (x_i^t x_j + 1)^p$$

  - Gaussian radial Basis kernel

$$K(x_i, x_j) = \exp\left(-\frac{1}{2\sigma^2}\|x_i - x_j\|^2\right)$$

  - Hyperbolic tangent (sigmoid) kernel

$$K(x_i,x_j) = \tanh(k\, x_i^t x_j + c)$$



- The mappings $\boldsymbol{\phi}(\mathbf{x}_i)$ never have to be computed!!

SVM Summary

- Advantages:
  - Based on very strong theory
  - Excellent generalization properties
  - Objective function has no local minima
  - Can be used to find non linear discriminant functions
  - Complexity of the classifier is characterized by the number of support vectors rather than the dimensionality of the transformed space
- Disadvantages:
  - Directly applicable to two-class problems
  - Quadratic programming is computationally expensive
  - Need to choose kernel

Conclusion

The support-vector network combines 3 ideas:

1. The solution technique from optimal hyperplanes (that allows for an expansion of the solution vector on support vectors).

2. The idea of convolufion of the dot-product (that extends the solution surfaces from linear to non-linear),

3. The notion of soft margins (to allow for errors on the training set).