Support vector machine (II): non-linear SVM

LING 572

Fei Xia

Week 8: 2/26/2013

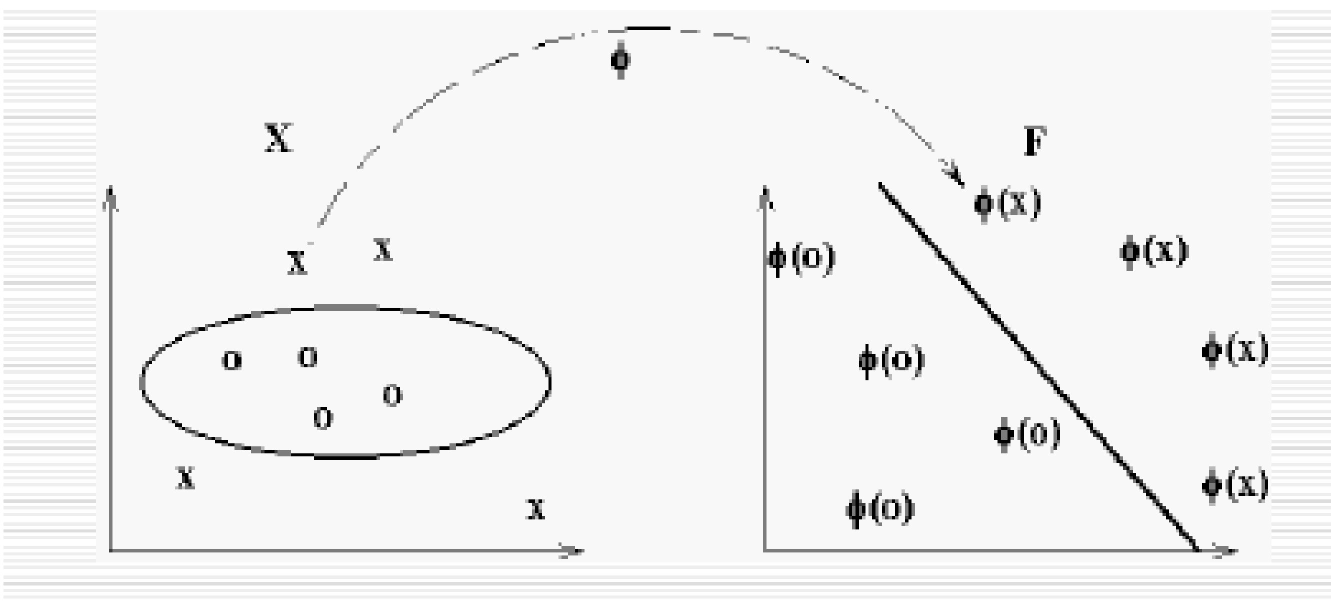
Outline

- Linear SVM
  - Maximizing the margin
  - Soft margin
- Nonlinear SVM
  - Kernel trick
- A case study
- Handling multi-class problems

Non-linear SVM

The highlight

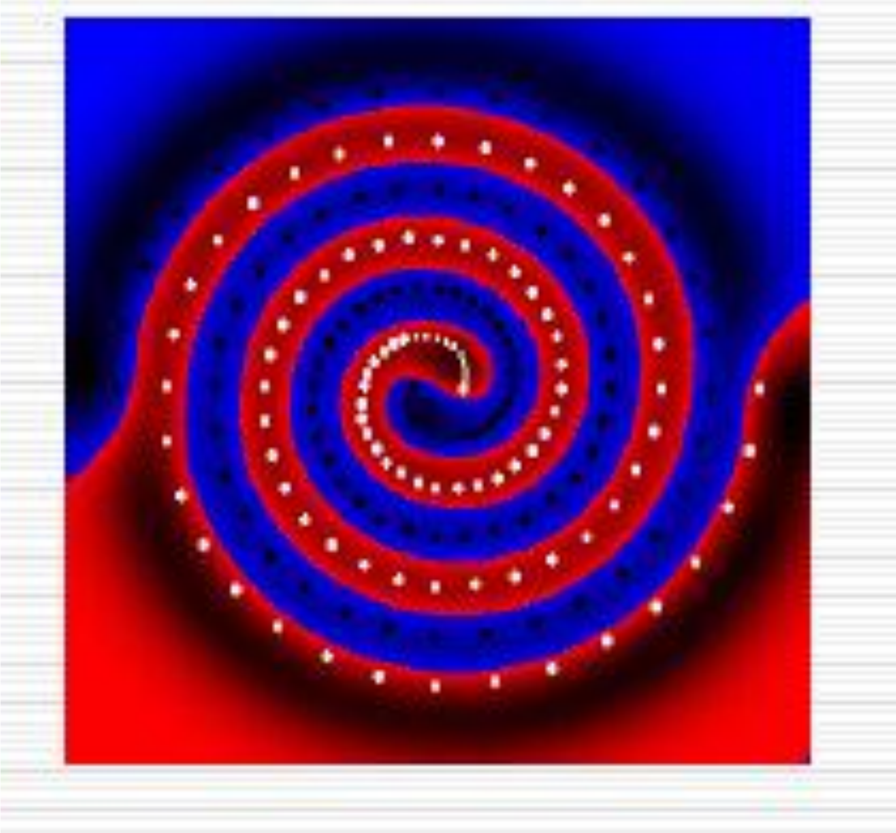
- Problem: Some data are not linear separable.
- Intuition: to transform the data to a high dimension space



Input space Feature space

Example: the two spirals

Separated by a hyperplane
in feature space (Gaussian
kernels)



Feature space

- Learning a non-linear classifier using SVM:
  - Define ϕ
  - Calculate ϕ(x) for each training example
  - Find a linear SVM in the feature space.

- Problems:
  - Feature space can be high dimensional or even have infinite dimensions.
  - Calculating ϕ(x) is very inefficient and even impossible.
  - Curse of dimensionality

Kernels

- Kernels are similarity functions that return inner products between the images of data points.

$$K\colon\ X \times X \to R$$
$$K(\vec{x}, \vec{z}) =< \phi(\vec{x}), \phi(\vec{z}) >$$

- Kernels can often be computed efficiently even for very high dimensional spaces.

- Choosing K is equivalent to choosing ϕ.

  ➔ the feature space is implicitly defined by K

An example

Let $\phi(\vec{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$

Let $\vec{x}$=(1,2) $\vec{z}$=(-2,3)

$\phi(\vec{x}) = (1, 4, 2\sqrt{2})$ $\phi(\vec{z}) = (4, 9, -6\sqrt{2})$

$$K(\vec{x}, \vec{z}) =< \phi(\vec{x}), \phi(\vec{z}) >$$

$$=< (1, 4, 2\sqrt{2}), (4, 9, -6\sqrt{2}) >$$

$$= 1 * 4 + 4 * 9 - 2 * 6 * 2 = 16$$

$$< \vec{x}, \vec{z} >= -2 + 2 * 3 = 4$$

An example**

$$\text{Let } \phi(\vec{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\boxed{K(\vec{x}, \vec{z})}$$

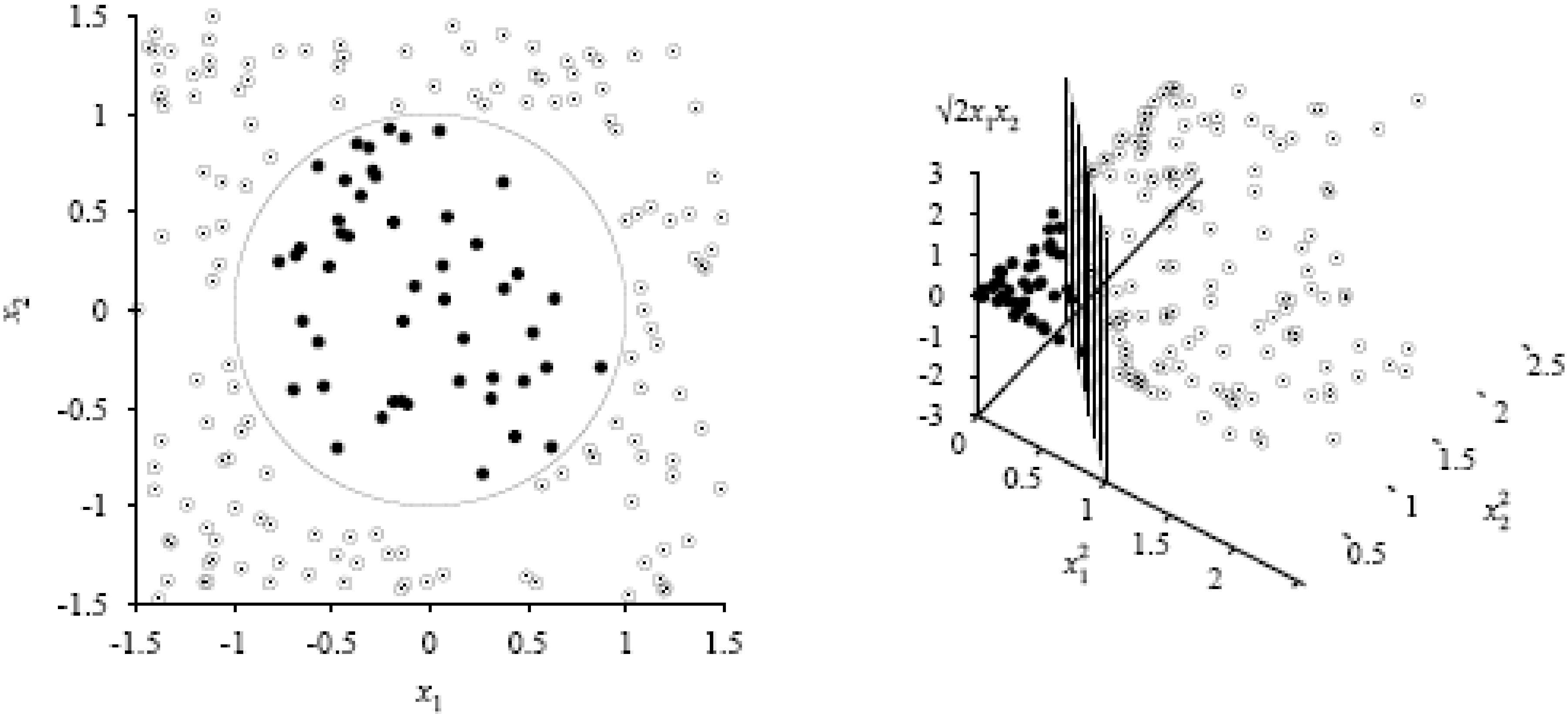
$$= < \phi(\vec{x}), \phi(\vec{z}) >$$

$$= < (x_1^2, x_2^2, \sqrt{2}x_1x_2), (z_1^2, z_2^2, \sqrt{2}z_1z_2) >$$

$$= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1z_1x_2z_2$$

$$= (x_1z_1 + x_2z_2)^2$$

$$= \boxed{< \vec{x}, \vec{z} >^2}$$



From Page 750 of (Russell and Norvig, 2002)

Another example**

$$\text{Let } \phi(\vec{x}) = (x_1^3, x_2^3, \sqrt{3}x_1^2 x_2, \sqrt{3}x_1 x_2^2)$$

$$K(\vec{x}, \vec{z})$$

$$=< \phi(\vec{x}), \phi(\vec{z}) >$$

$$=< (x_1^3, x_2^3, \sqrt{3}x_1^2 x_2, \sqrt{3}x_1 x_2^2), (z_1^3, z_2^3, \sqrt{3}z_1^2 z_2, \sqrt{3}z_1 z_2^2) >$$

$$= x_1^3 z_1^3 + x_2^3 z_2^3 + 3x_1^2 z_1^2 x_2 z_2 + 3x_1 z_1 x_2^2 z_2^2$$

$$= (x_1 z_1 + x_2 z_2)^3$$

$$=< \vec{x}, \vec{z} >^3$$

The kernel trick

- No need to know what ϕ is and what the feature space is.
- No need to explicitly map the data to the feature space.
- Define a kernel function K, and replace the dot produce <x,z> with a kernel function K(x,z) in both training and testing.

Training

Maximize

$$L(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boxed{< \vec{x_i}, \vec{x_j} >}$$

Subject to $\alpha_i \geq 0 \text{ and } \sum_i \alpha_i y_i = 0$

Non-linear SVM

$$L(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boxed{K(\vec{x_i}, \vec{x_j})}$$

Decoding

Linear SVM: (without mapping)

$$f(\vec{x}) = < \vec{w}, \vec{x} > + b$$

$$= \sum_i \alpha_i y_i \boxed{< \vec{x_i}, \vec{x} >} + b$$

Non-linear SVM: w could be infinite dimensional

$$f(\vec{x}) = \sum_i \alpha_i y_i \boxed{K(\vec{x_i}, \vec{x})} + b$$

Kernel vs. features

Training: Maximize $L(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \boxed{K(\vec{x_i}, \vec{x_j})}$

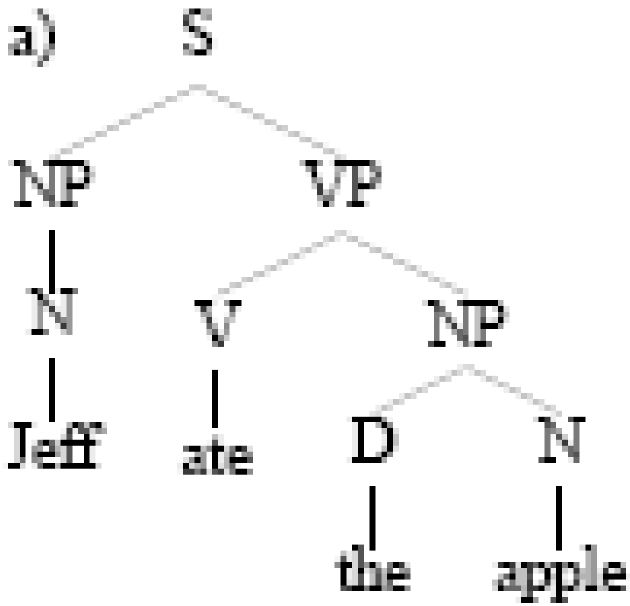
subject to $\alpha_i \geq 0$ and $\sum_i \alpha_i y_i = 0$

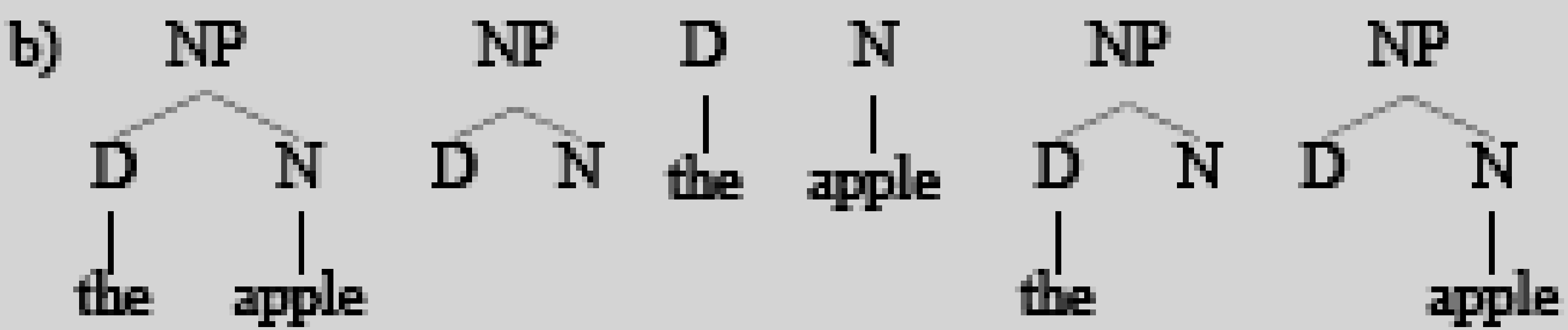
Decoding: $f(\vec{x}) = \sum_i \alpha_i y_i \boxed{K(\vec{x_i}, \vec{x})} + b$

Need to calculate K(x, z).

For some kernels, no need to represent x as a feature vector.

A tree kernel





Common kernel functions

- Linear : $K(\vec{x}, \vec{z}) =< \vec{x}, \vec{z} >$
- Polynominal: $K(\vec{x}, \vec{z}) = (\gamma < \vec{x}, \vec{z} > +c)^d$
- Radial basis function (RBF): $K(\vec{x}, \vec{z}) = e^{-\gamma(||\vec{x}-\vec{z}||)^2}$
- Sigmoid: $K(\vec{x}, \vec{z}) = tanh(\gamma < \vec{x}, \vec{z} > +c)$

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$||\vec{x} - \vec{z}||$$

$$x = (x_1, x_2, ..., x_n)$$

$$z = (z_1, z_2, ..., z_n)$$

$$\vec{x} - \vec{z} = (x_1 - z_1, ..., x_n - z_n)$$

$$||\vec{x} - \vec{z}|| = \sqrt{(x_1 - z_1)^2 + ... + (x_n - z_n)^2}$$

Polynomial kernel

- It allows us to model feature conjunctions (up to the order of the polynomial).

- Ex:
  - Original feature: single words
  - Quadratic kernel: word pairs, e.g., "ethnic" and "cleansing", "Jordan" and "Chicago"

Other kernels

- Kernels for
  - trees
  - sequences
  - sets
  - graphs
  - general structures
  - …

- A tree kernel example next time

The choice of kernel function

- Given a function, we can test whether it is a kernel function by using Mercer's theorem (see "Additional slides").

- Different kernel functions could lead to very different results.

- Need some prior knowledge in order to choose a good kernel.

Summary so far

- Find the hyperplane that maximizes the margin.
- Introduce soft margin to deal with noisy data
- Implicitly map the data to a higher dimensional space to deal with non-linear problems.
- The kernel trick allows infinite number of features and efficient computation of the dot product in the feature space.
- The choice of the kernel function is important.

MaxEnt vs. SVM

	MaxEnt	SVM
Modeling	Maximize P(Y|X, λ)	Maximize the margin
Training	Learn λ_i for each feature function	Learn α_i for each training instance and b
Decoding	Calculate P(y|x)	Calculate the sign of f(x). It is not prob
Things to decide	Features Regularization Training algorithm	Kernel Regularization Training algorithm Binarization

More info

- Website: www.kernel-machines.org

- Textbook (2000): www.support-vector.net

- Tutorials: http://www.svms.org/tutorials/

- Workshops at NIPS

Additional slides

Linear kernel

- The map ϕ is linear.

$$\phi(x) = (a_1 x_1, a_2 x_2, ..., a_n x_n)$$

$$\begin{aligned} K(x,z) &= < \phi(x), \phi(z) > \\ &= a_1^2 x_1 z_1 + a_2^2 x_2 z_2 + ... + a_n^2 x_n z_n \end{aligned}$$

- The kernel adjusts the weight of the features according to their importance.

The Kernel Matrix (a.k.a. the Gram matrix)

K(1,1)	K(1,2)	K(1,3)	…	K(1,m)
K(2,1)	K(2,2)	K(2,3)	…	K(2,m)
…				
…				
K(m,1)	K(m,2)	K(m,3)	…	K(m,m)

K(i, j) means K(x_i, x_j),

where x_i means the i-th training instance.

Mercer’s Theorem

- The kernel matrix is symmetric positive definite.

- Any symmetric, positive definite matrix can be regarded as a kernel matrix; that is, there exists a ϕ such that $K(x, z) = <\phi(x), \phi(z)>$

Making kernels

- The set of kernels is closed under some operations. For instance, if K_1 and K_2 are kernels, so are the following:
  - $K_1 + K_2$
  - cK_1 and cK_2 for $c > 0$
  - $cK_1 + dK_2$ for $c > 0$ and $d > 0$

- One can make complicated kernels from simples ones