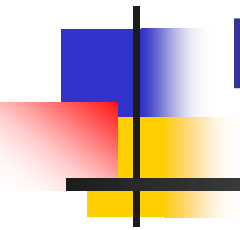
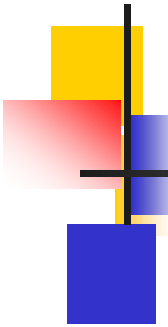


An Introduction to Support Vector Machines



Outline

- History of support vector machines (SVM)
- Two classes, linearly separable
  - What is a good decision boundary?
- Two classes, not linearly separable
- How to make SVM non-linear: kernel trick
- Conclusion

Pattern Analysis

- Three properties:
  - Computational efficiency
    - The performance of the algorithm scales to large datasets.
  - Robustness
    - Insensitivity of the algorithm to noise in the training examples
  - Statistical Stability
    - The detected regularities should indeed be patterns of the underlying source

History

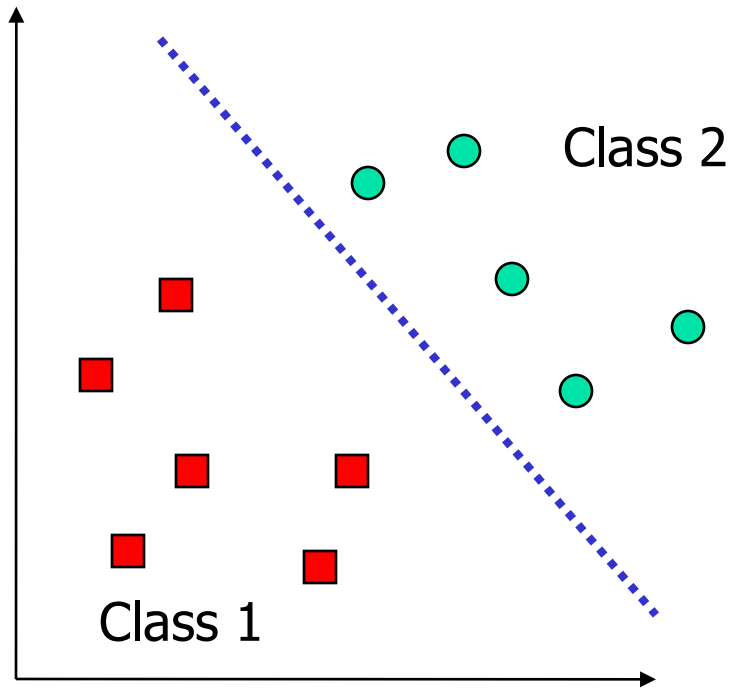
- The mathematical result underlying the kernel trick, Mercer's theorem, is almost a century old (Mercer 1909). It tells us that any 'reasonable' kernel function corresponds to some feature space.

- The underlying mathematical results that allow us to determine which kernels can be used to compute distances in feature spaces was developed by Schoenberg (1938).

History of SVM

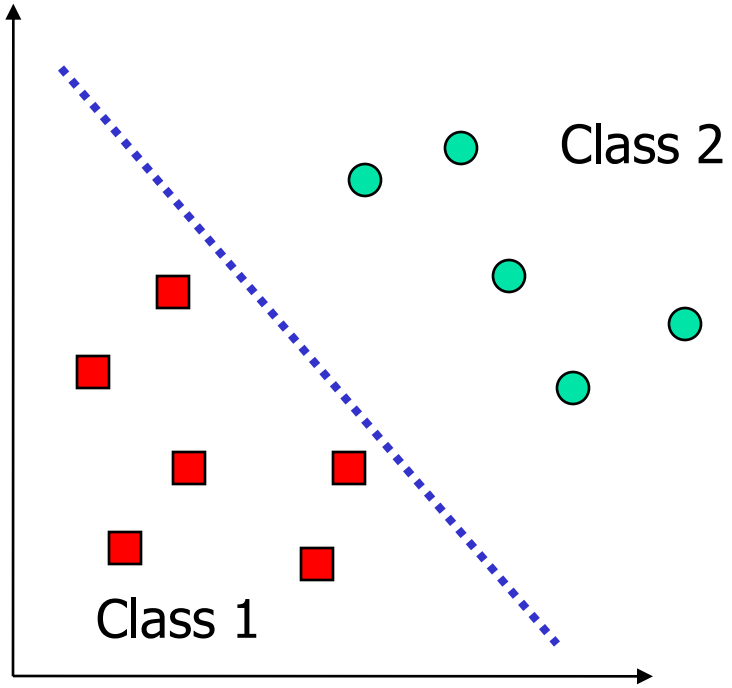
- SVM is a classifier derived from statistical learning theory by Vapnik and Chervonenkis
- SVM was first introduced in COLT-92
- SVM becomes famous when, using pixel maps as input, it gives accuracy comparable to sophisticated neural networks with elaborated features in a handwriting recognition task
- Currently, SVM is closely related to:
  - Kernel methods, large margin classifiers, reproducing kernel Hilbert space, Gaussian process

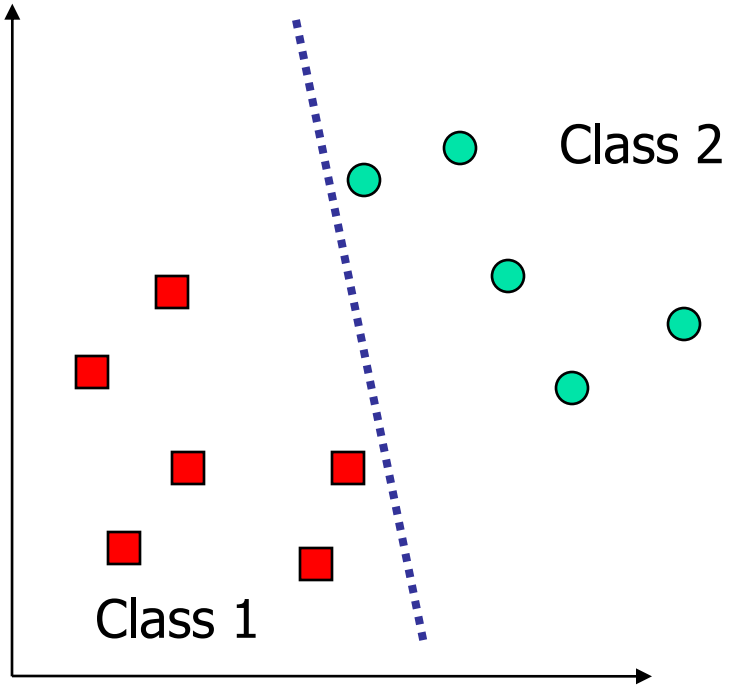
Two Class Problem: Linear Separable Case



- Many decision boundaries can separate these two classes
- Which one should we choose?

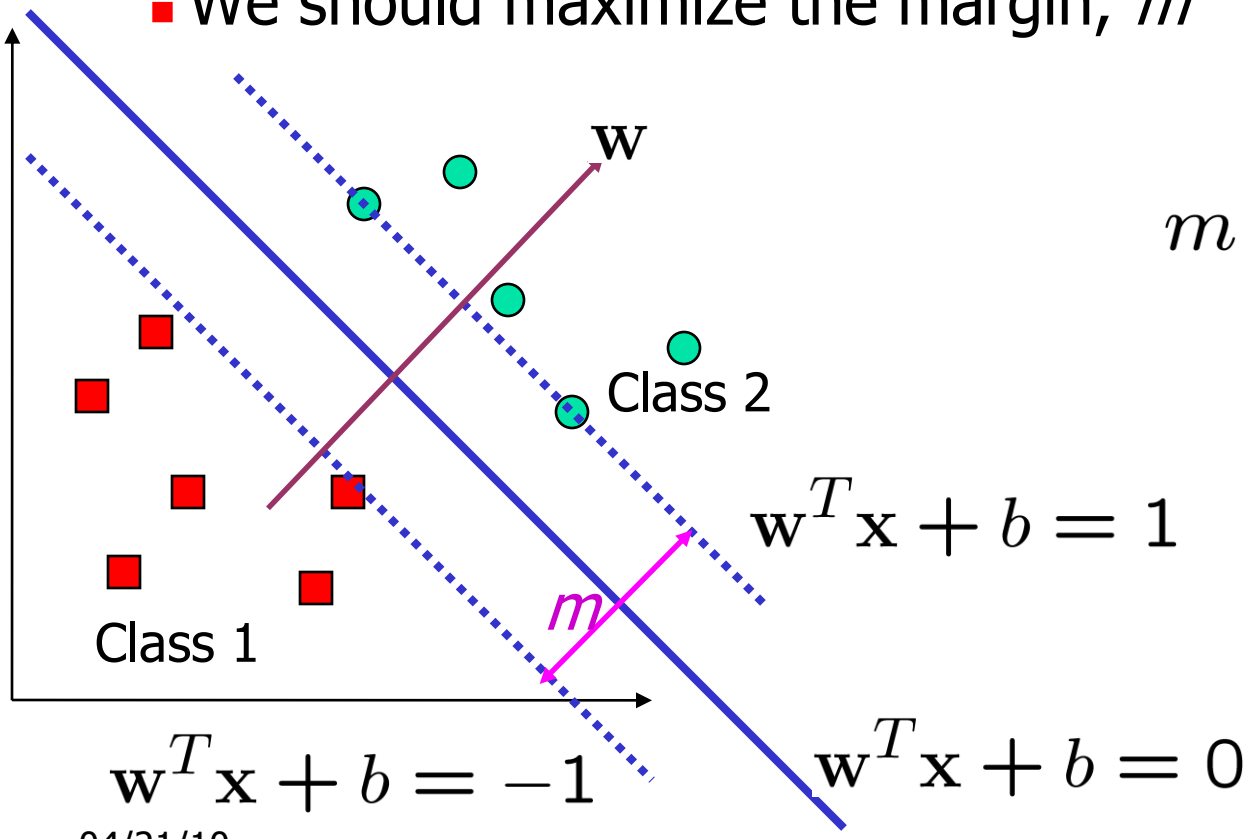
Example of Bad Decision Boundaries





Good Decision Boundary: Margin Should Be Large

- The decision boundary should be as far away from the data of both classes as possible
  - We should maximize the margin, *m*



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

The Optimization Problem

- Let $\{x_1, ..., x_n\}$ be our data set and let $y_i \in \{1,-1\}$ be the class label of x_i
- The decision boundary should classify all points correctly $\Rightarrow \; y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \qquad \forall i$
- A constrained optimization problem

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

The Optimization Problem

- We can transform the problem to its dual

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

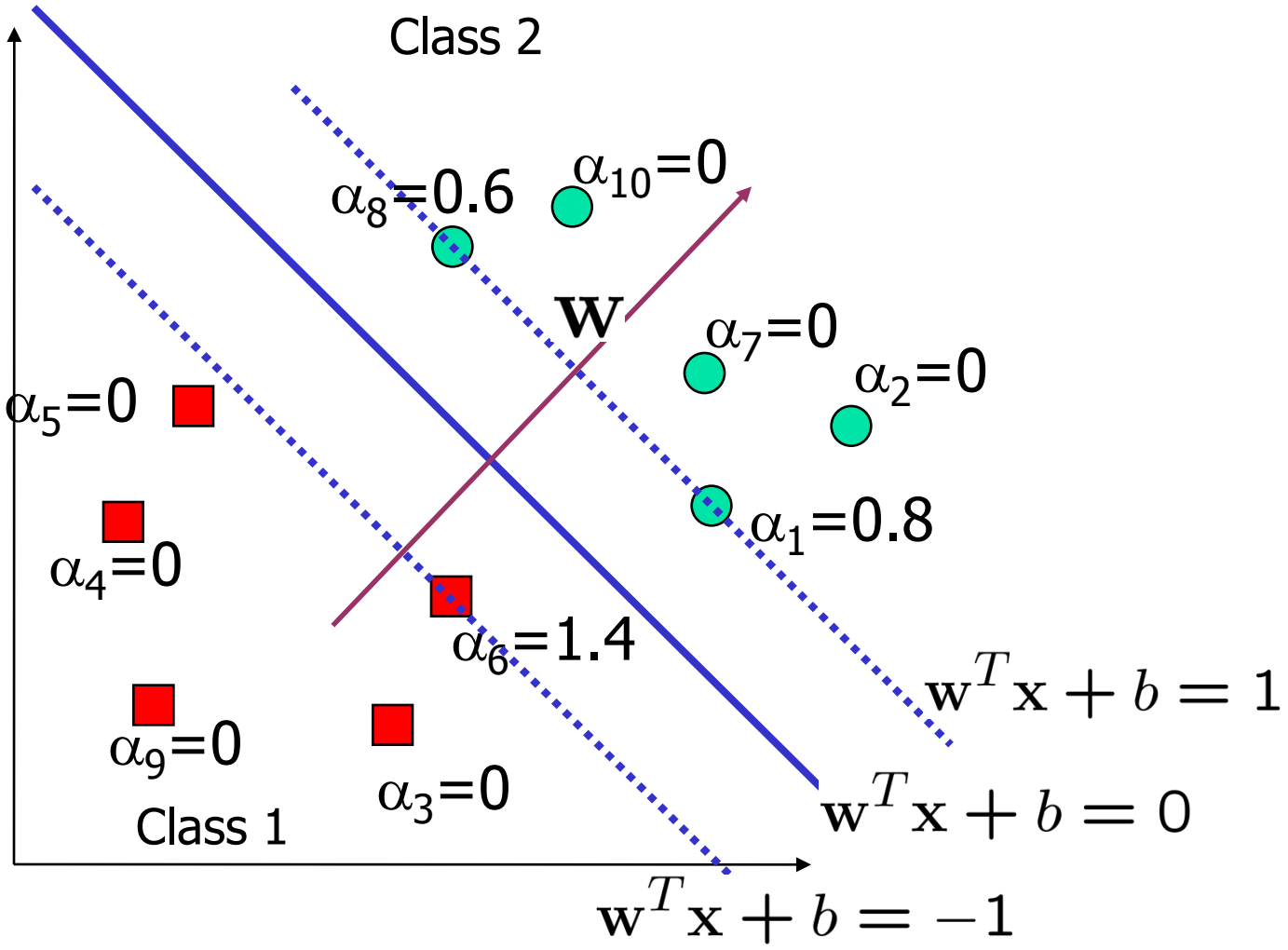
$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

- This is a quadratic programming (QP) problem
  - Global maximum of α_i can always be found
- **w** can be recovered by $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$

Characteristics of the Solution

- Many of the α_i are zero
  - **w** is a linear combination of a small number of data
  - Sparse representation
- $\mathbf{x}_i$ with non-zero α_i are called support vectors (SV)
  - The decision boundary is determined only by the SV
  - Let t_j (j=1, ..., s) be the indices of the s support vectors. We can write $\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$
- For testing with a new data **z**
  - Compute $\mathbf{w}^T\mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} (\mathbf{x}_{t_j}^T \mathbf{z}) + b$ and classify **z** as class 1 if the sum is positive, and class 2 otherwise

A Geometrical Interpretation

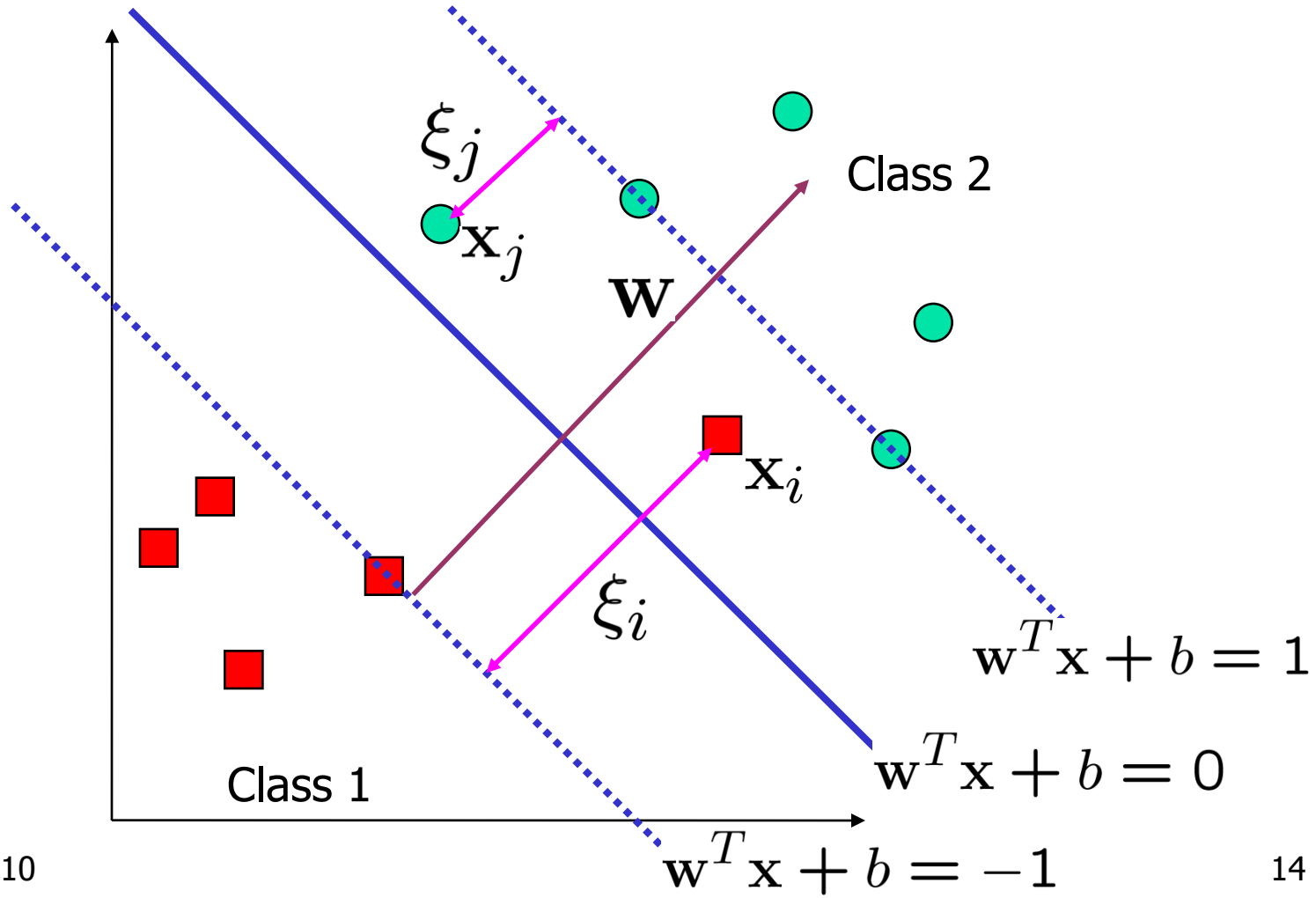


Some Notes

- There are theoretical upper bounds on the error on unseen data for SVM
  - The larger the margin, the smaller the bound
  - The smaller the number of SV, the smaller the bound
- Note that in both training and testing, the data are referenced only as inner product, $\mathbf{x}^T\mathbf{y}$
  - This is important for generalizing to the non-linear case

How About Not Linearly Separable

- We allow "error" ξ_i in classification



Soft Margin Hyperplane

- Define ξ_i=0 if there is no error for x_i
  - ξ_i are just "slack variables" in optimization theory

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

- We want to minimize $\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$
  - C: tradeoff parameter between error and margin
- The optimization problem becomes

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

The Optimization Problem

- The dual of the problem is

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

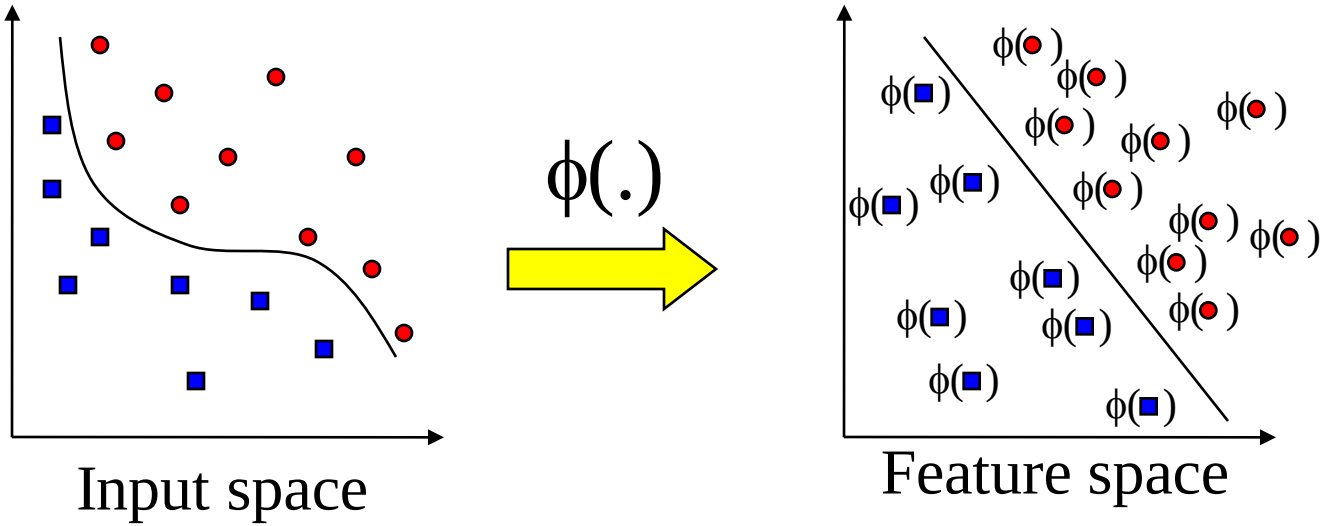
- **w** is also recovered as $\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$
- The only difference with the linear separable case is that there is an upper bound *C* on α_i
- Once again, a QP solver can be used to find α_i

Extension to Non-linear Decision Boundary

- Key idea: transform $\mathbf{x}_i$ to a higher dimensional space to "make life easier"
  - Input space: the space $\mathbf{x}_i$ are in
  - Feature space: the space of $\phi(\mathbf{x}_i)$ after transformation
- Why transform?
  - Linear operation in the feature space is equivalent to non-linear operation in input space
  - The classification task can be "easier" with a proper transformation. Example: XOR

Extension to Non-linear Decision Boundary

- Possible problem of the transformation
  - High computation burden and hard to get a good estimate
- SVM solves these two issues simultaneously
  - Kernel tricks for efficient computation
  - Minimize $||\mathbf{w}||^2$ can lead to a "good" classifier



Example Transformation

- Define the kernel function $K(\mathbf{x},\mathbf{y})$ as

$$K(\mathbf{x},\mathbf{y}) = (1 + x_1y_1 + x_2y_2)^2$$

- Consider the following transformation

$$\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) = (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1y_2)$$

$$\langle \phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}), \phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) \rangle = (1 + x_1y_1 + x_2y_2)^2$$

$$= K(\mathbf{x},\mathbf{y})$$

- The inner product can be computed by K without going through the map $\phi(.)$

Kernel Trick

- The relationship between the kernel function *K* and the mapping $\phi(.)$ is

$$K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$$

  - This is known as the kernel trick
- In practice, we specify *K*, thereby specifying $\phi(.)$ indirectly, instead of choosing $\phi(.)$
- Intuitively, *K* (**x**,**y**) represents our desired notion of similarity between data **x** and **y** and this is from our prior knowledge
- *K* (**x**,**y**) needs to satisfy a technical condition (Mercer condition) in order for $\phi(.)$ to exist

Examples of Kernel Functions

- Polynomial kernel with degree *d*

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^d$$

- Radial basis function kernel with width σ

$$K(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/(2\sigma^2))$$

  - Closely related to radial basis function neural networks

- Sigmoid with parameter κ and θ

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa\mathbf{x}^T\mathbf{y} + \theta)$$

  - It does not satisfy the Mercer condition on all κ and θ

- Research on different kernel functions in different applications is very active

Multi-class Classification

- SVM is basically a two-class classifier
- One can change the QP formulation to allow multi-class classification
- More commonly, the data set is divided into two parts "intelligently" in different ways and a separate SVM is trained for each way of division
- Multi-class classification is done by combining the output of all the SVM classifiers
  - Majority rule
  - Error correcting code
  - Directed acyclic graph

Software

- A list of SVM implementation can be found at http://www.kernel-machines.org/software.html
- Some implementation (such as LIBSVM) can handle multi-class classification
- SVMLight is among one of the earliest implementation of SVM
- Several Matlab toolboxes for SVM are also available

Summary: Steps for Classification

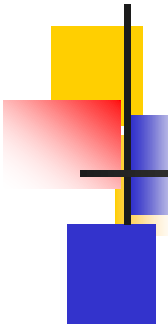
- Prepare the pattern matrix
- Select the kernel function to use
- Select the parameter of the kernel function and the value of *C*
  - You can use the values suggested by the SVM software, or you can set apart a validation set to determine the values of the parameter
- Execute the training algorithm and obtain the α_i
- Unseen data can be classified using the α_i and the support vectors

Strengths and Weaknesses of SVM

- Strengths
  - Training is relatively easy
    - No local optimal, unlike in neural networks
  - It scales relatively well to high dimensional data
  - Tradeoff between classifier complexity and error can be controlled explicitly
  - Non-traditional data like strings and trees can be used as input to SVM, instead of feature vectors
- Weaknesses
  - Need a "good" kernel function

Other Types of Kernel Methods

- A lesson learnt in SVM: a linear algorithm in the feature space is equivalent to a non-linear algorithm in the input space
- Classic linear algorithms can be generalized to its non-linear version by going to the feature space
  - Kernel principal component analysis, kernel independent component analysis, kernel canonical correlation analysis, kernel k-means, 1-class SVM are some examples



Conclusion

- SVM is a useful alternative to neural networks
- Two key concepts of SVM: maximize the margin and the kernel trick
- Many active research is taking place on areas related to SVM
- Many SVM implementations are available on the web for you to try on your data set!

Resources

- http://www.kernel-machines.org/
- http://www.support-vector.net/
- http://www.support-vector.net/icml-tutorial.pdf
- http://www.kernel-machines.org/papers/tutorial-nips.ps.gz
- http://www.clopinet.com/isabelle/Projects/SVM/applist.html

KERNEL METHODS

History

ANOVA kernels were first suggested by Burges and Vapnik (1995) (under the name Gabor kernels).

- Schölkopf, Smola and Müller (1996) used kernel functions to perform principal component analysis.

- Schölkopf (1997) observed that any algorithm which can be formulated solely in terms of dot products can be made non-linear by carrying it out in feature spaces induced by Mercer kernels. Schölkopf, Smola and Müller (1997) presented their paper on kernel PCA.

Overview

- **Kernel Methods:** New class of pattern analysis algorithms
  - can operate on very general types of data
  - can detect very general types of relations.

- A powerful and principled way of detecting nonlinear relations using **well-understood linear algorithms** in an **appropriate feature space.**

Kernel Trick

- **Kernel trick:** Using a **linear classifier** algorithm to solve a **non-linear problem** by mapping the original non-linear observations into a **higher-dimensional space**
  - Linear classification in the new space equivalent to non-linear classification in the original space

- **Mercer's theorem:** Any continuous, symmetric, positive semi-definite(i.e eigenvalues are positive) kernel function K(x, y) can be expressed as a **dot product** in a high-dimensional space.

Embed Data into A Feature Space

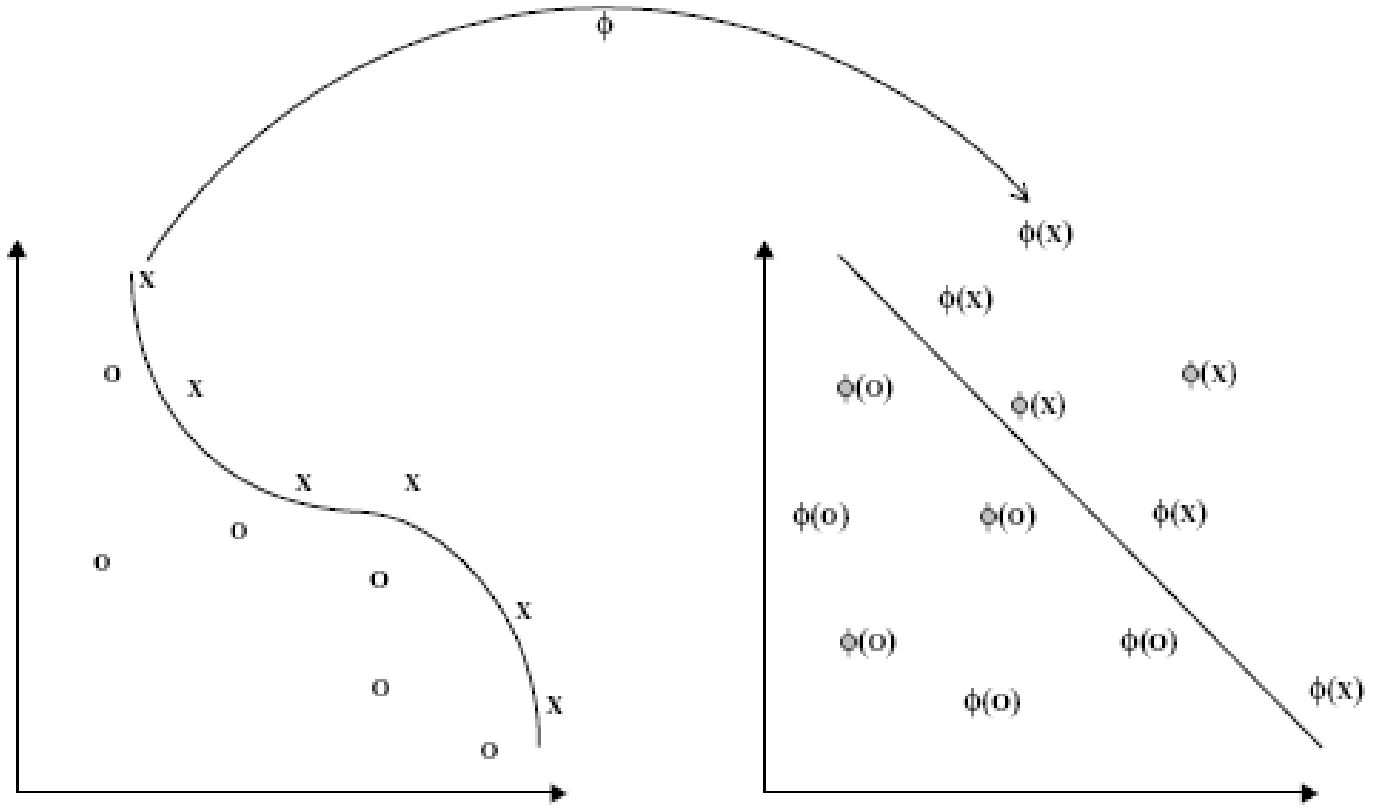
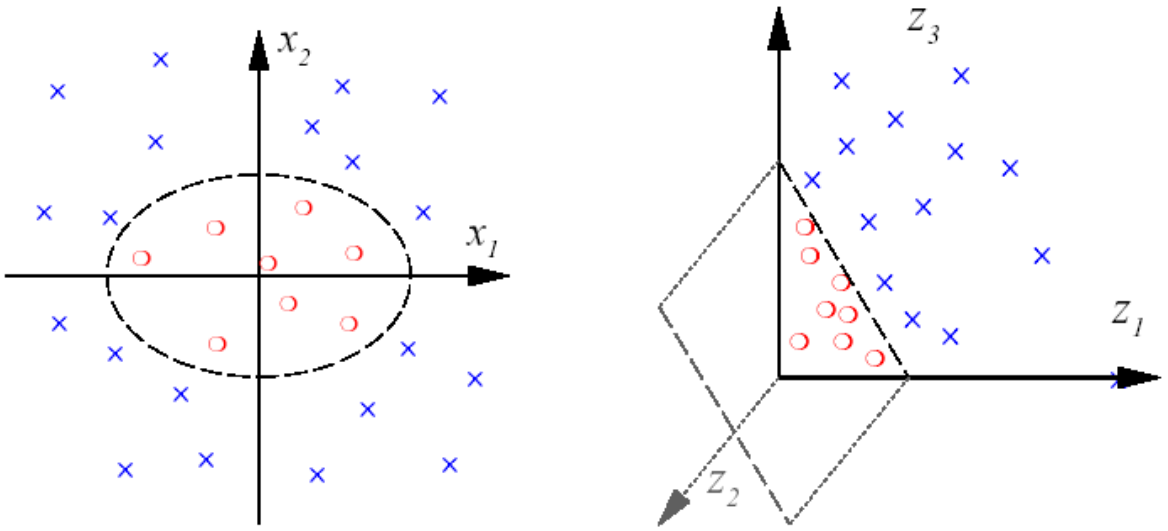


Fig. 2.1. The function ϕ embeds the data into a feature space where the nonlinear pattern now appears linear. The kernel computes inner products in the feature space directly from the inputs.

MOTIVATION

- Linearly inseparable problems become linearly separable in higher dimension space

$$\Phi : \mathbb{R}^2 \to \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$$



B. Schölkopf, NIPS, 3 December 2001

Kernel function

- Kernel Function: A function that returns the **inner product** between the images of **two inputs in some feature space**.
  - $K(x1,x2)= <\varphi(x1),\varphi(x2)>$

- Choosing K is equivalent to choosing Φ (the embedding map)

An example-Polinomial Kernel

$$x = (x_1, x_2);$$

$$z = (z_1, z_2);$$

$$\langle x, z \rangle^2 = (x_1 z_1 + x_2 z_2)^2 =$$

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

$$= \left\langle (x_1^2, x_2^2, \sqrt{2} x_1 x_2), (z_1^2, z_2^2, \sqrt{2} z_1 z_2) \right\rangle =$$

$$= \langle \phi(x), \phi(z) \rangle$$

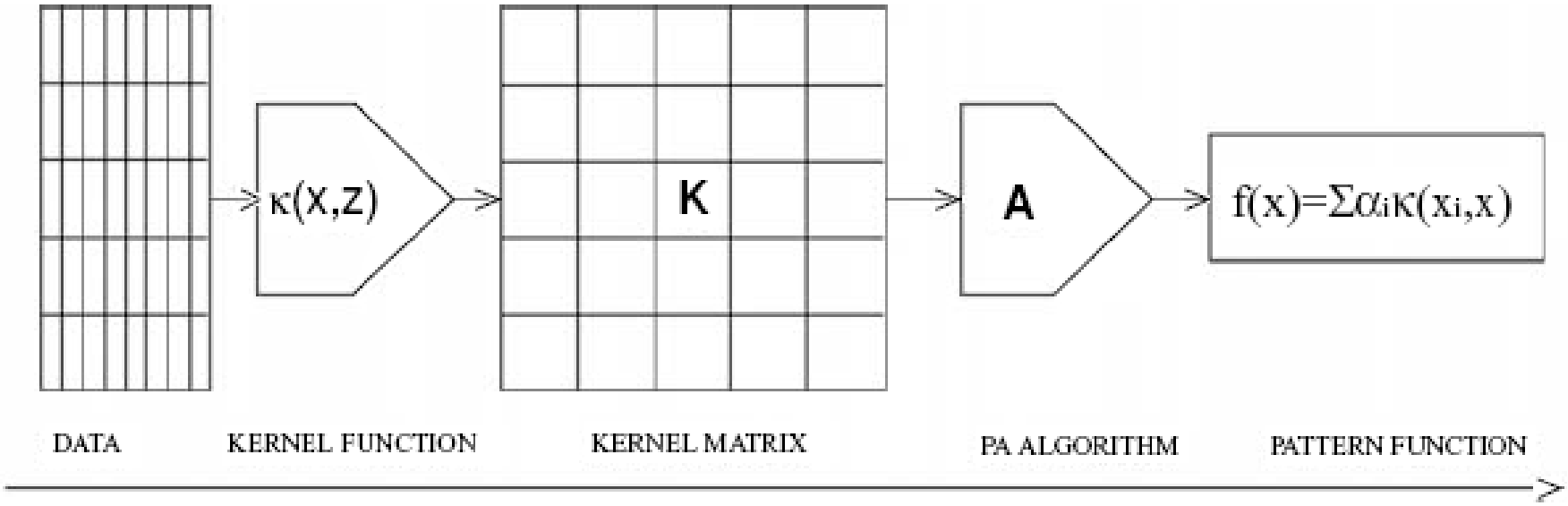
Common Kernel Functions

Gaussian RBF	$k(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{-\|\mathbf{x} - \mathbf{y}\|^2}{c}\right)$
Polynomial	$((\mathbf{x} \cdot \mathbf{y}) + \theta)^d$
Sigmoidal	$\tanh(\kappa(\mathbf{x} \cdot \mathbf{y}) + \theta)$
inv. multiquadric	$\frac{1}{\sqrt{\|\mathbf{x} - \mathbf{y}\|^2 + c^2}}$

Stages in Kernel Methods

- Embed the data in a **suitable** feature space
  - Kernel functions
    - Depend on the specific data type and domain knowledge
- Use algorithm based on linear algebra, geometry and statistics to discover patterns in embedded data.
  - The pattern analysis component
    - General purpose and robust

Stages in Kernel Methods(cont)



- First → Create kernel matrix using kernel function
- Second → Apply pattern analysis algorithm

The kernel matrix

$\mathbf{K}$	1	2	$\cdots$	ℓ
1	$\kappa(\mathbf{x}_1, \mathbf{x}_1)$	$\kappa(\mathbf{x}_1, \mathbf{x}_2)$	$\cdots$	$\kappa(\mathbf{x}_1, \mathbf{x}_\ell)$
2	$\kappa(\mathbf{x}_2, \mathbf{x}_1)$	$\kappa(\mathbf{x}_2, \mathbf{x}_2)$	$\cdots$	$\kappa(\mathbf{x}_2, \mathbf{x}_\ell)$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
ℓ	$\kappa(\mathbf{x}_\ell, \mathbf{x}_1)$	$\kappa(\mathbf{x}_\ell, \mathbf{x}_2)$	$\cdots$	$\kappa(\mathbf{x}_\ell, \mathbf{x}_\ell)$

- Symmetric Positive Definite (positive eigenvalues)
- Contains **all necessary** information for the learning algorithm

A Universal Kernel?

- Universal kernel is not possible

  - The kernels must be chosen for the problem.

Kernel Types

- Polynomial Kernels
- Gaussian Kernels
- ANOVA Kernels
- Kernels from Graphs
- Kernels on Sets
- Kernels on Real Numbers
- Randomized kernels
- Kernels for text
- Kernels for structured data: Strings, Trees, etc.

Kernels

Polynomial kernel
All-subsets kernel
Gaussian kernel
ANOVA kernel
Alternative recursion for ANOVA kernel
General graph kernels
Exponential diffusion kernel
von Neumann diffusion kernel
Evaluating diffusion kernels
Evaluating randomised kernels
Intersection kernel
Union-complement kernel
Agreement kernel
Derived subsets kernel
Kernels on real numbers
Spline kernels
Vector space kernel
Latent semantic kernels
The p-spectrum kernel
The p-spectrum recursion
Blended spectrum kernel
All-subsequences kernel
Fixed length subsequences kernel
Naive recursion for gap-weighted subsequences kernel
Gap-weighted subsequences kernel
Trie-based string kernels

ANOVA kernel
Simple graph kernels
All–non-contiguous subsequences kernel
Fixed length subsequences kernel
Gap-weighted subsequences kernel
Character weighting string kernel
Soft matching string kernel
Gap number weighting string kernel
Trie-based p-spectrum kernel
Trie-based mismatch kernel
Trie-based restricted gap-weighted kernel
Co-rooted subtree kernel
All-subtree kernel
Fixed length HMM kernel
Pair HMM kernel
Hidden tree model kernel
Fixed length Markov model Fisher kernel

Pattern Analysis Methods

- Supervised Learning
  - Support Vector Machines
  - Kernel Fisher Discriminant
- Unsupervised Learning
  - Kernel PCA
  - Kernel k-means

Applications

- Geostatistics
  - Analysis of mining processes through mathematical models
- Bioinformatics
  - Application of information technology to the field of molecular biology
- Cheminformatics
  - Use of computer and informational techniques in the field of chemistry.
- Text categorization
  - Assign an electronic document to one or more categories, based on its contents
- Handwriting Recognition
- Speech Recognition

REFERENCES

- John Shawe-Taylor, Nello Cristianini(2004). Kernel Methods for Pattern Analysis,Cambridge University Press
- Klaus-Robert Müller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda, and Bernhard Schölkopf.(2001) An Introduction to Kernel-Based Learning Algorithms. IEEE Transactıons On Neural Networks, Vol. 12, No. 2, March 2001
- Nello Cristianini. Kernel Methods for General Pattern Analysis. Access:15.12.2008. *www.kernel-methods.net/tutorials/KMtalk.pdf*