CS 4700: Foundations of Artificial Intelligence

Carla P. Gomes
gomes@cs.cornell.edu

Module:
SVM
(Reading: Chapter 20.6)

Adapted from Martin Law's slides
http://www.henrykautz.org/ai/intro_SVM_new.ppt

Support Vector Machines (SVM)

- Supervised learning methods for classification and regression

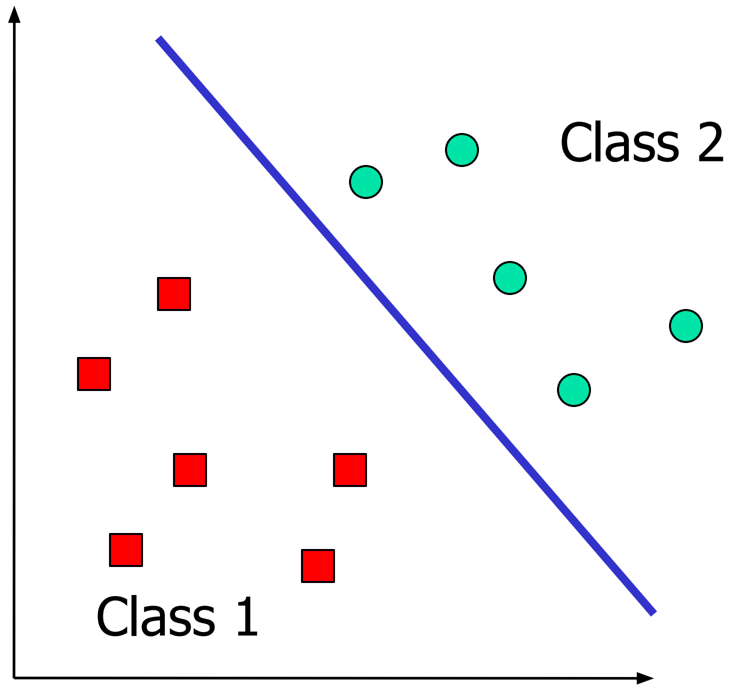
  relatively new class of successful learning methods -

- they can represent non-linear functions and they have an efficient training algorithm

- derived from statistical learning theory by Vapnik and Chervonenkis (COLT-92)

- SVM got into mainstream because of their exceptional performance in Handwritten Digit Recognition
  - 1.1% error rate which was comparable to a very carefully constructed (and complex) ANN

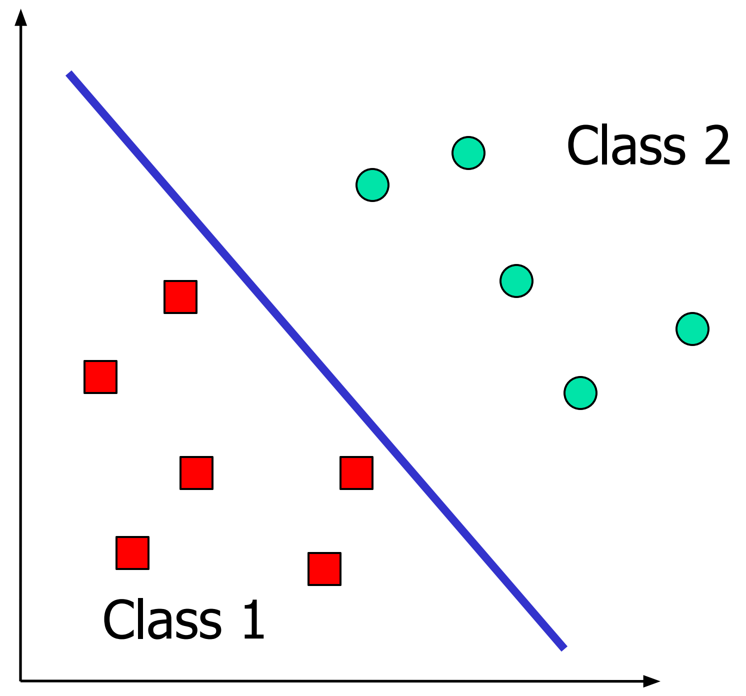
Two Class Problem: Linear Separable Case

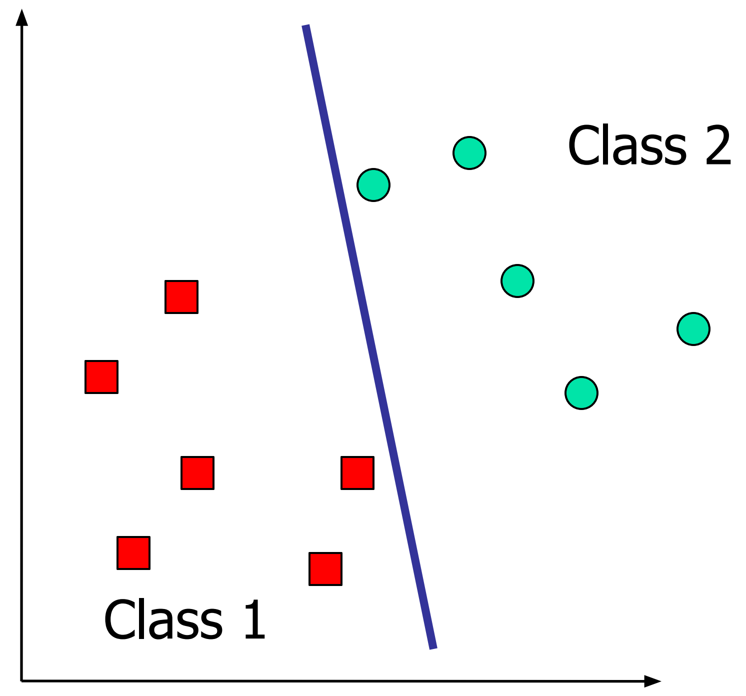


Many decision boundaries can separate these two classes

Which one should we choose?

Example of Bad Decision Boundaries





Good Decision Boundary: Margin Should Be Large

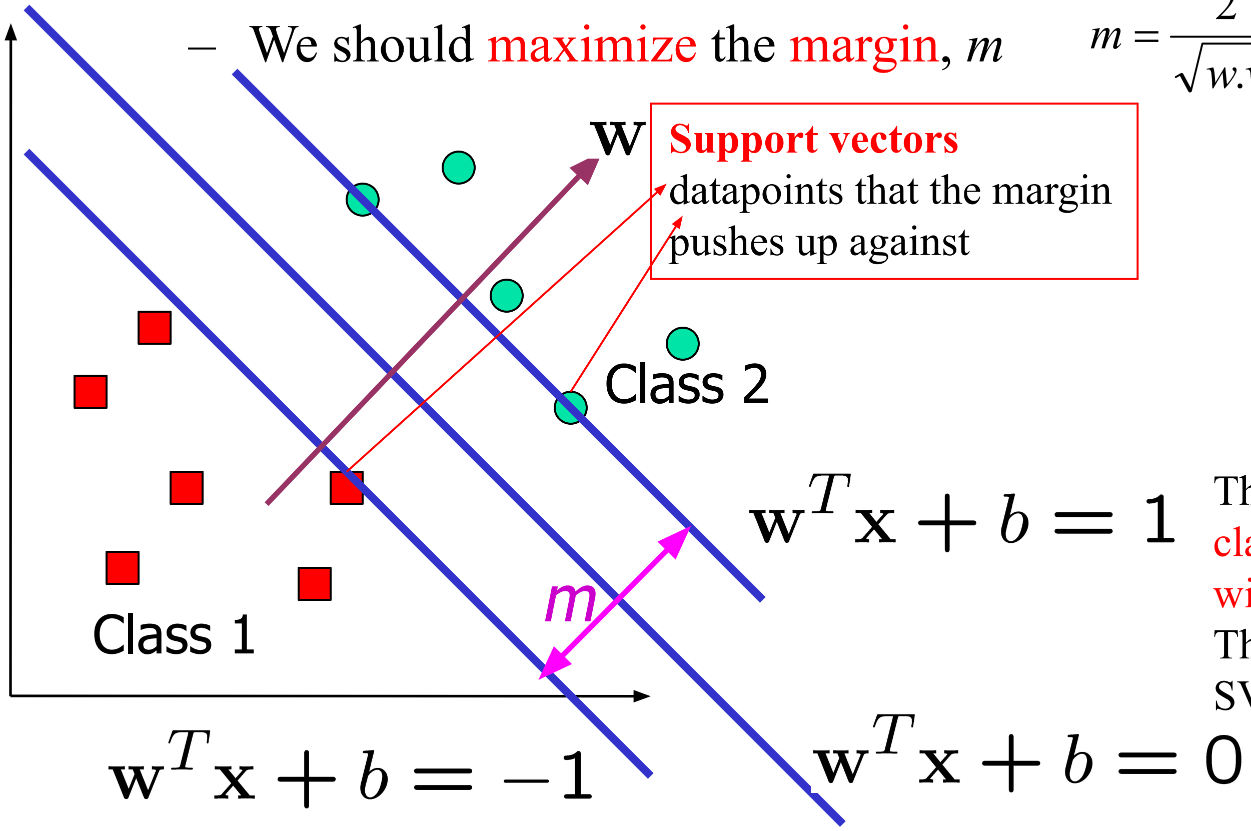
The decision boundary should be as far away from the data of both classes as possible

– We should maximize the margin, m

$$m = \frac{2}{\sqrt{w.w}}$$

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

$$\|\mathbf{x}\| := \sqrt{x_1^2 + \cdots + x_n^2}.$$



$\mathbf{w}$
Class 2
Class 1
m
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}^T\mathbf{x} + b = -1$
$\mathbf{w}^T\mathbf{x} + b = 0$

Support vectors
datapoints that the margin pushes up against

The maximum margin linear classifier is the linear classifier with the maximum margin. This is the simplest kind of SVM (Called an Linear SVM)

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 $\Rightarrow$

A constrained optimization problem

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

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

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

- $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$

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

Lagrangian of Original Problem

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

$$\text{subject to } 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) \leq 0 \qquad \text{for } i = 1, \ldots, n$$

The Lagrangian is

Lagrangian multipliers

$$\mathcal{L} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} \alpha_i \left(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\right)$$

– Note that $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$

Setting the gradient of $\mathcal{L}$ w.r.t. **w** and b to zero, we have

$$\mathbf{w} + \sum_{i=1}^{n} \alpha_i(-y_i)\mathbf{x}_i = \mathbf{0} \quad \Rightarrow \quad \boxed{\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i}$$

$$\boxed{\sum_{i=1}^{n} \alpha_i y_i = 0} \qquad \alpha_i \geq 0$$

The Dual Optimization Problem

We can transform the problem to its dual

Dot product of X

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

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

α's □ New variables (Lagrangian multipliers)

This is a convex quadratic programming (QP) problem

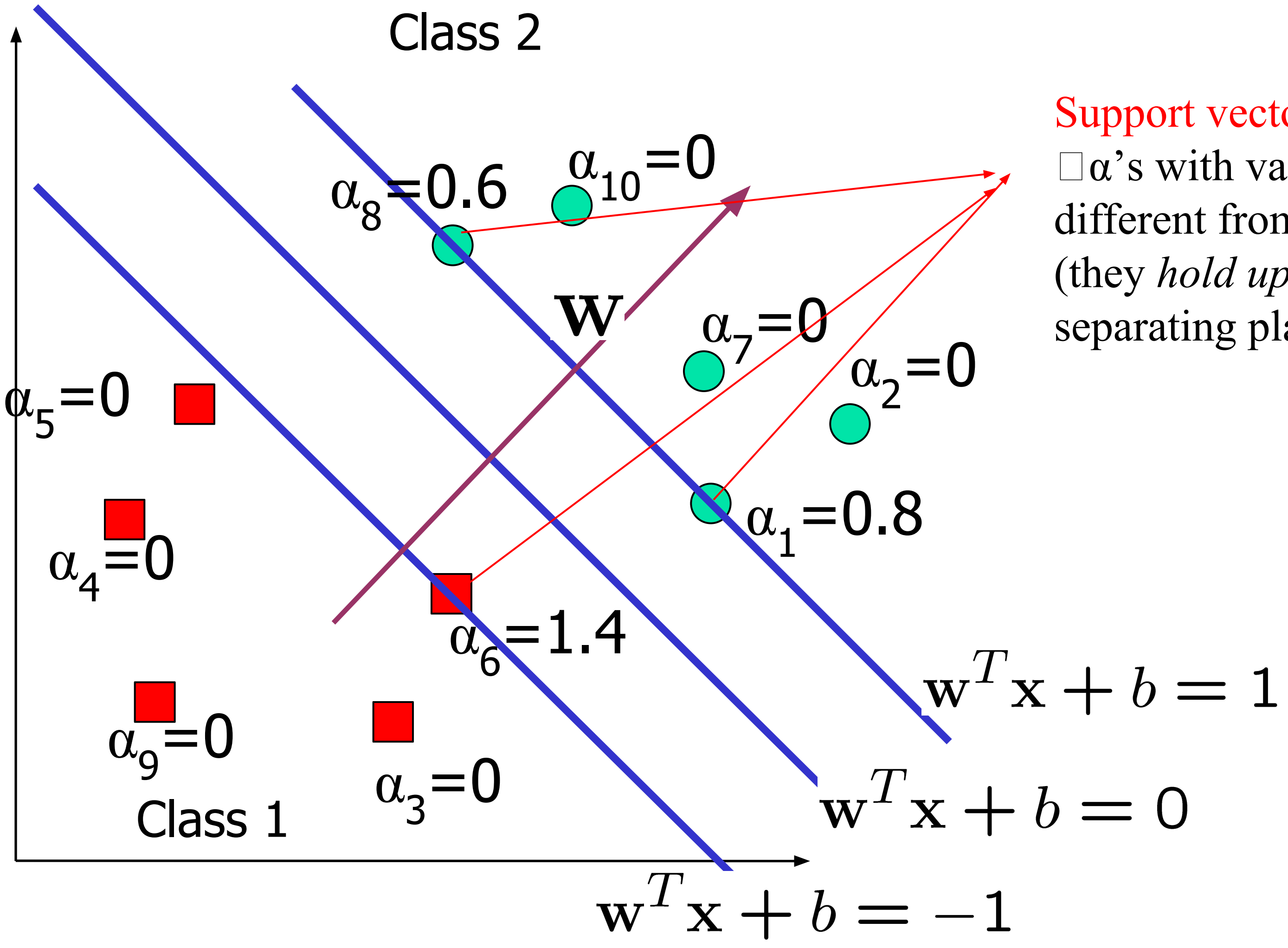
– Global maximum of α_i can always be found

□well established tools for solving this optimization problem (e.g. cplex)

Note:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

A Geometrical Interpretation

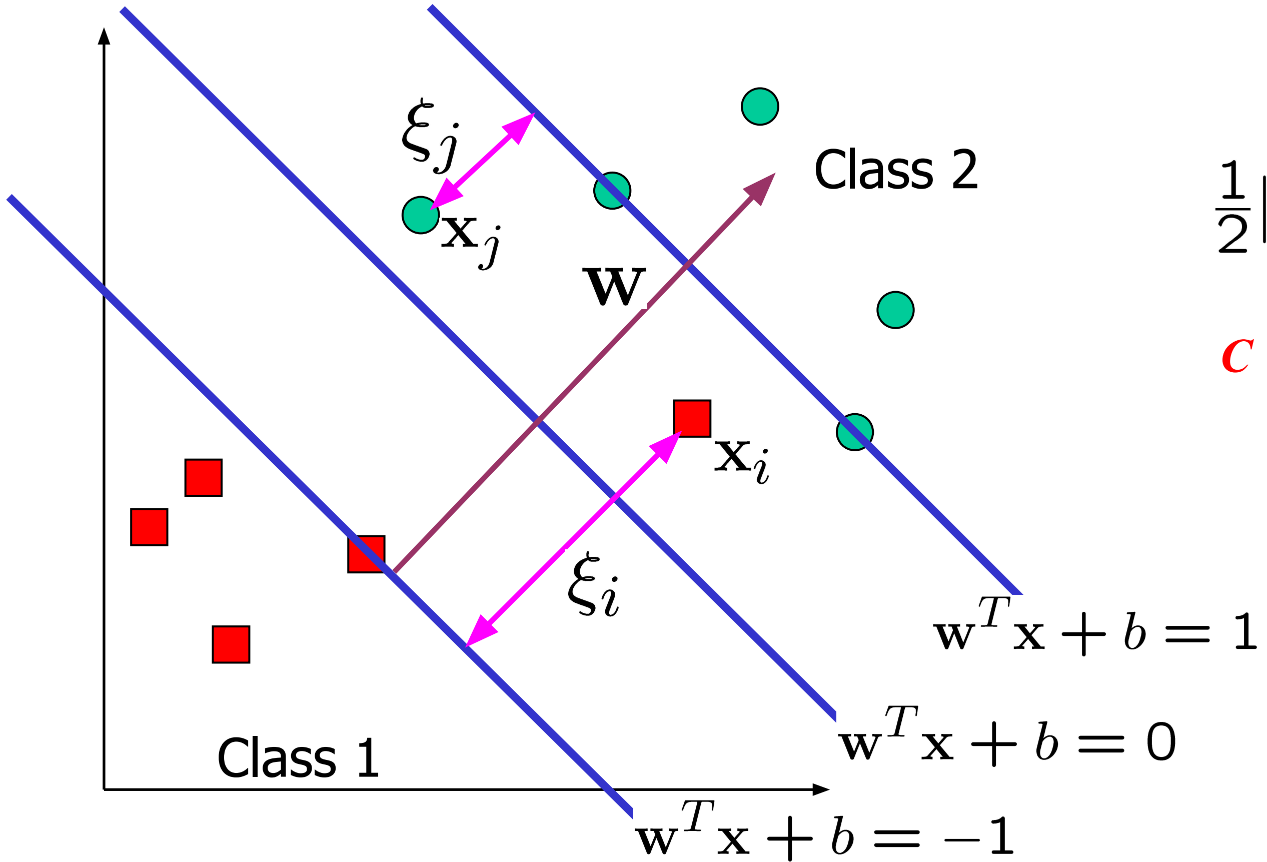


Support vectors
□α's with values different from zero (they *hold up* the separating plane)!

Non-linearly Separable Problems

We allow “error” ξ_i in classification; it is based on the output of the discriminant function $\mathbf{w}^T\mathbf{x}+b$

ξ_i approximates the number of misclassified samples



ξ_j
$\mathbf{x}_j$
Class 2
$\mathbf{w}$
$\mathbf{x}_i$
ξ_i
$\mathbf{w}^T\mathbf{x}+b=1$
$\mathbf{w}^T\mathbf{x}+b=0$
$\mathbf{w}^T\mathbf{x}+b=-1$
Class 1

New objective function:

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

C : tradeoff parameter between error and margin; chosen by the user; large C means a higher penalty to errors

The Optimization Problem

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

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

$$\mathbf{w} = \textstyle\sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

w is also recovered as

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 efficiently!!!

Extension to Non-linear SVMs (Kernel Machines)

Non-Linear SVM

How could we generalize this procedure to non-linear data?

Vapnik in 1992 showed that transforming input data $\mathbf{x}_i$ into a higher dimensional makes the problem easier.

Similar to Hidden Layers in ANN

– We know that data appears only as dot products $(\mathbf{x}_i.\mathbf{x}_j)$

– Suppose we transform the data to some (possibly infinite dimensional) space **H** via a mapping function Φ such that the data appears of the form $\Phi(\mathbf{x}_i)\Phi(\mathbf{x}_j)$

Why?

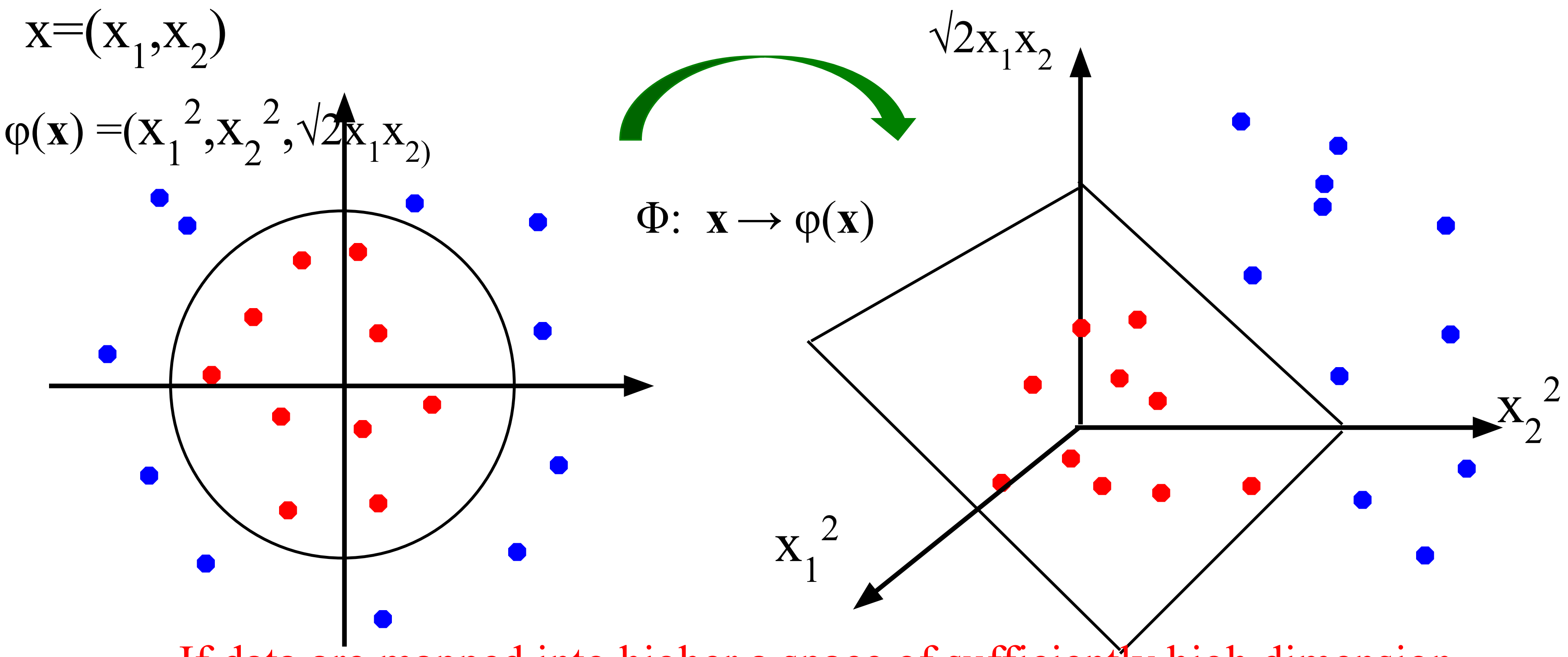
– Linear operation in **H** is equivalent to non-linear operation in input space.

Non-linear SVMs: Feature Space

General idea: the original input space (x) can be mapped to some higher-dimensional feature space ($\varphi(\mathbf{x})$)where the training set is separable:

$x=(x_1,x_2)$

$\varphi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$



If data are mapped into higher a space of sufficiently high dimension, then they will in general be linearly separable;
N data points are in general separable in a space of N-1 dimensions or more!!!

This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

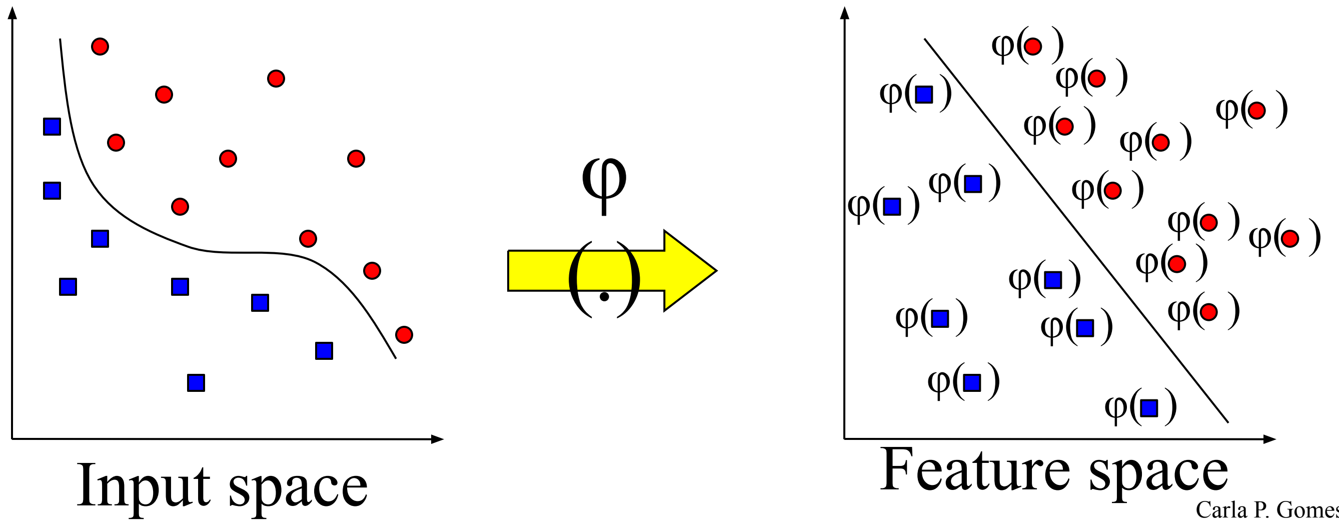
Transformation to Feature Space

Possible problem of the transformation

- High computation burden due to high-dimensionality 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



Kernel Trick ☺

Recall:

Note that data only appears as dot products

maximize $$\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=j=1}^{N}\alpha_i\alpha_j y_i y_j \boxed{x_i x_j}$$

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

Since data is only represented as dot products, we need not do the mapping explicitly.

Introduce a Kernel Function (*) *K* such that:

$$K(x_i, x_j) = \phi(x_i)\cdot\phi(x_j)$$

(*)Kernel function – a function that can be applied to pairs of input data to evaluate dot products in some corresponding feature space

Example Transformation

Consider the following transformation

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$
$$\phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) = (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1y_2)$$

Define the kernel function *K* (**x**,**y**) as

$$\begin{aligned}\left\langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right)\right\rangle &= (1 + x_1y_1 + x_2y_2)^2 \\ &= K(\mathbf{x}, \mathbf{y})\end{aligned}$$

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

The inner product φ(.)φ(.) can be computed by *K* without going through the map φ(.) explicitly!!!

Modification Due to Kernel Function

Change all inner products to kernel functions

For training,

Original

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

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

With kernel function

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

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

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

Example

Suppose we have 5 1D data points

- $x_1=1$, $x_2=2$, $x_3=4$, $x_4=5$, $x_5=6$, with 1, 2, 6 as class 1 and 4, 5 as class 2 $\Rightarrow$ $y_1=1$, $y_2=1$, $y_3=-1$, $y_4=-1$, $y_5=1$

We use the polynomial kernel of degree 2

- $K(x,y) = (xy+1)^2$
- C is set to 100

We first find α_i (i=1, …, 5) by

$$\text{max. } \sum_{i=1}^{5} \alpha_i - \frac{1}{2} \sum_{i=1}^{5} \sum_{j=1}^{5} \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$$

$$\text{subject to } 100 \geq \alpha_i \geq 0, \sum_{i=1}^{5} \alpha_i y_i = 0$$

Example

By using a QP solver, we get

$\alpha_1=0, \alpha_2=2.5, \alpha_3=0, \alpha_4=7.333, \alpha_5=4.833$

- Verify (at home) that the constraints are indeed satisfied
- The support vectors are $\{x_2=2, x_4=5, x_5=6\}$

The discriminant function is

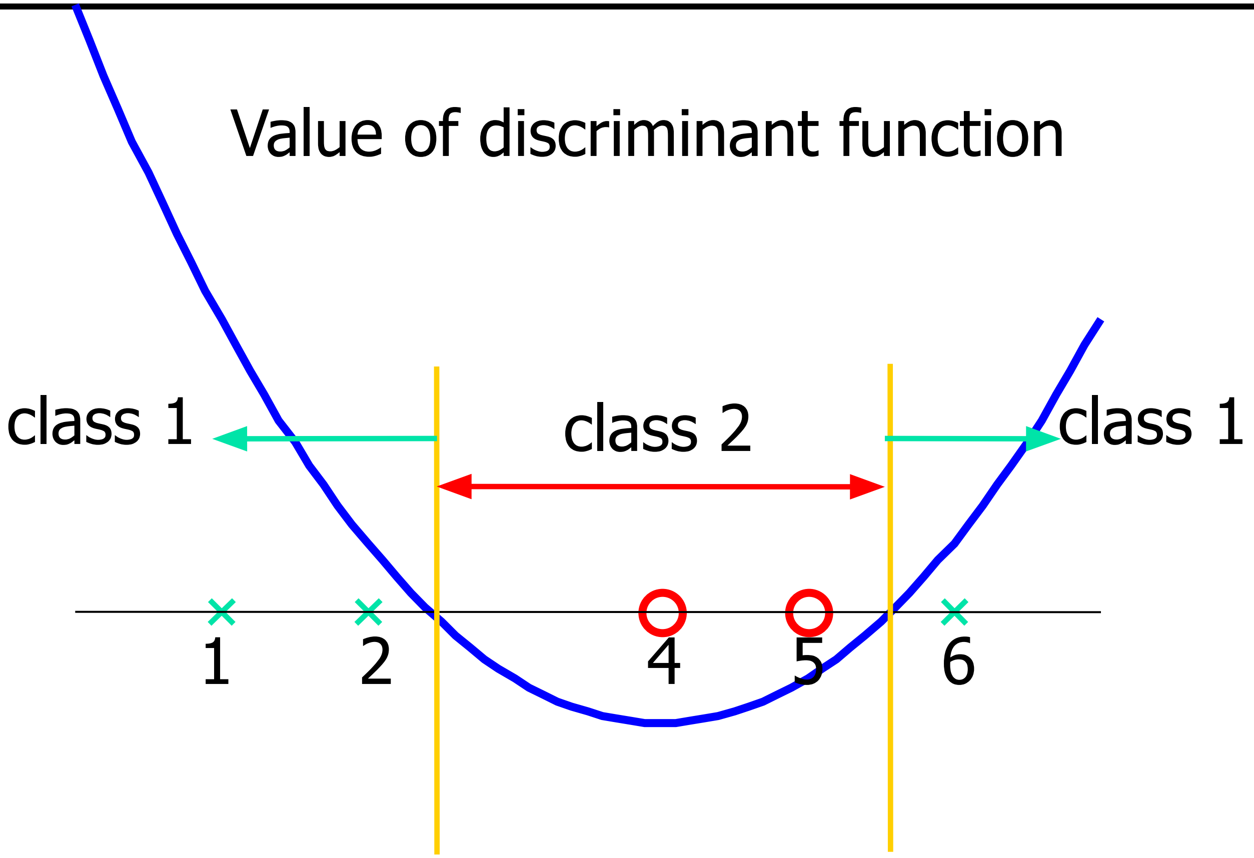
$$\begin{aligned} f(y) &= 2.5(1)(2y+1)^2 + 7.333(-1)(5y+1)^2 + 4.833(1)(6y+1)^2 + b \\ &= 0.6667x^2 - 5.333x + b \end{aligned}$$

b is recovered by solving f(2)=1 or by f(5)=-1 or by f(6)=1, as x_2, x_4, x_5 lie on and all give b=9

$$y_i(\mathbf{w}^T\phi(z)+b)=1$$

$$\Longrightarrow \quad f(y) = 0.6667x^2 - 5.333x + 9$$

Example


Value of discriminant function
class 1
class 2
class 1
1
2
4
5
6

Choosing the Kernel Function

Probably the most tricky part of using SVM.

The kernel function is important because it creates the kernel matrix, which summarizes all the data

Many principles have been proposed (diffusion kernel, Fisher kernel, string kernel, …)

There is even research to estimate the kernel matrix from available information

In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try

Note that SVM with RBF kernel is closely related to RBF neural networks, with the centers of the radial basis functions automatically chosen for SVM

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

Recap of Steps in SVM

Prepare data matrix $\{(x_i, y_i)\}$
Select a Kernel function
Select the error parameter *C*
“Train” the system (to find all α_i)
New data can be classified using α_i and Support Vectors

Summary

Weaknesses

- Training (and Testing) is quite slow compared to ANN
  - Because of Constrained Quadratic Programming
- Essentially a binary classifier
  - However, there are some tricks to evade this.
- Very sensitive to noise
  - A few off data points can completely throw off the algorithm
- Biggest Drawback: The choice of Kernel function.
  - There is no “set-in-stone” theory for choosing a kernel function for any given problem (still in research...)
  - Once a kernel function is chosen, there is only ONE modifiable parameter, the error penalty C.

Summary

Strengths

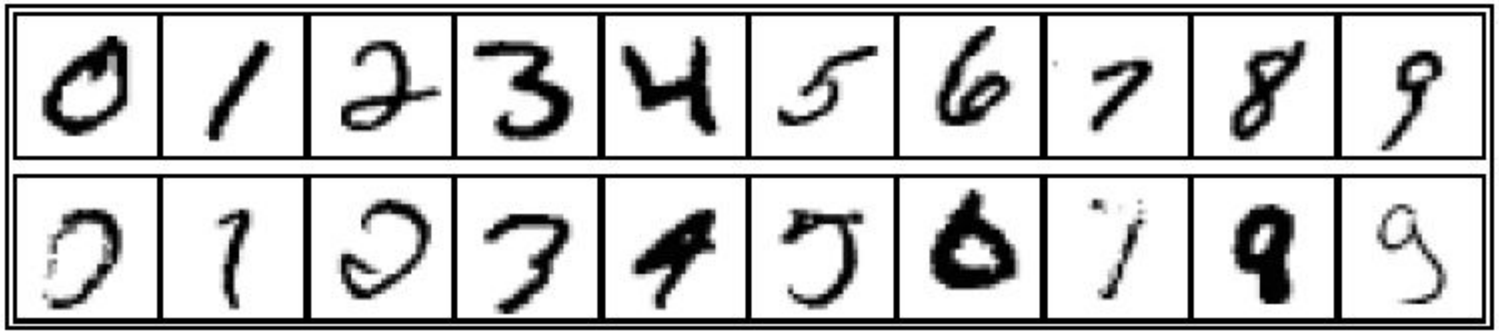
- Training is relatively easy
  - We don't have to deal with local minimum like in ANN
  - SVM solution is always global and unique (check "Burges" paper for proof and justification).
- Unlike ANN, doesn't suffer from "curse of dimensionality".
  - How? Why? We have infinite dimensions?!
  - Maximum Margin Constraint: DOT-PRODUCTS!
- Less prone to overfitting
- Simple, easy to understand geometric interpretation.
  - No large networks to mess around with.

Applications of SVMs

- Bioinformatics
- Machine Vision
- Text Categorization
- Ranking (e.g., Google searches) $\leftarrow$ Prof. Throsten Joachims
- Handwritten Character Recognition
- Time series analysis

□Lots of very successful applications!!!

Handwritten digit recognition



3-nearest-neighbor = 2.4% error

400–300–10 unit MLP = 1.6% error

LeNet: 768–192–30–10 unit MLP = 0.9% error

Current best (kernel machines, vision algorithms) ≈ 0.6% error

References

Burges, C. “A Tutorial on Support Vector Machines for Pattern Recognition.” Bell Labs. 1998

Law, Martin. “A Simple Introduction to Support Vector Machines.” Michigan State University. 2006

Prabhakar, K. “An Introduction to Support Vector Machines”

Resources

http://www.kernel-machines.org

http://www.support-vector.net/

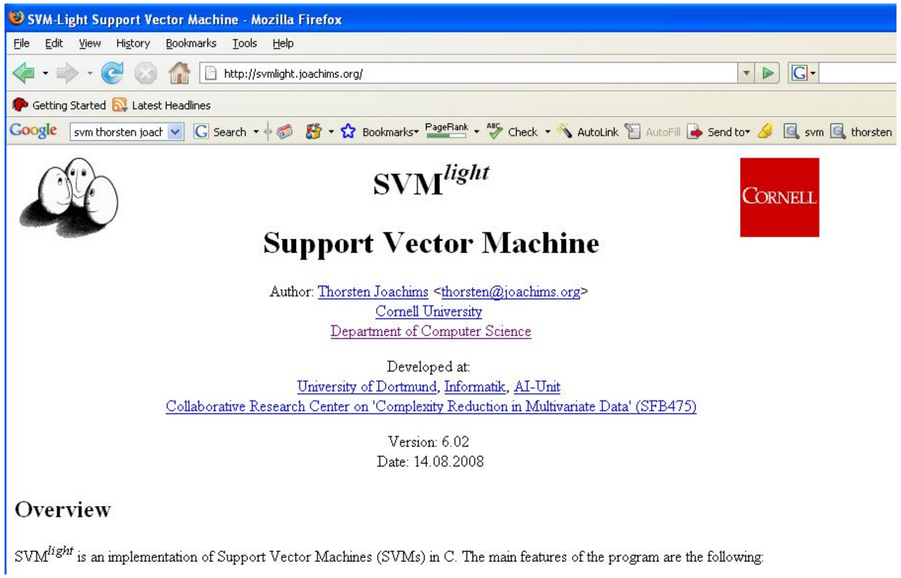
http://www.support-vector.net/icml-tutorial.pdf

http://www.kernel-machines.org/papers/tutorial-nips.ps.gz

http://www.clopinet.com/isabelle/Projects/SVM/applist.html

http://www.cs.cornell.edu/People/tj/

http://svmlight.joachims.org/


SVM-Light Support Vector Machine - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://svmlight.joachims.org/
Getting Started Latest Headlines
Google svm thorsten joach Search Bookmarks PageRank Check AutoLink AutoFill Send to svm thorsten
CORNELL

SVM^{light}

Support Vector Machine

Author: Thorsten Joachims <thorsten@joachims.org>
Cornell University
Department of Computer Science

Developed at:
University of Dortmund, Informatik, AI-Unit
Collaborative Research Center on 'Complexity Reduction in Multivariate Data' (SFB475)

Version: 6.02
Date: 14.08.2008

Overview

SVM^{light} is an implementation of Support Vector Machines (SVMs) in C. The main features of the program are the following: