

# Support Vector Machines

Greg Grudic

(Notes borrowed from Bernhard Schölkopf)

## Today's Lecture Goals

- Support Vector Machine Classification

A Good text on SVMs: Bernhard Schölkopf and Alex Smola. **Learning with Kernels**. MIT Press, Cambridge, MA, 2002

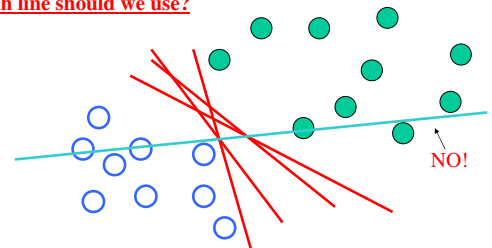
## Support Vector Machine Classification

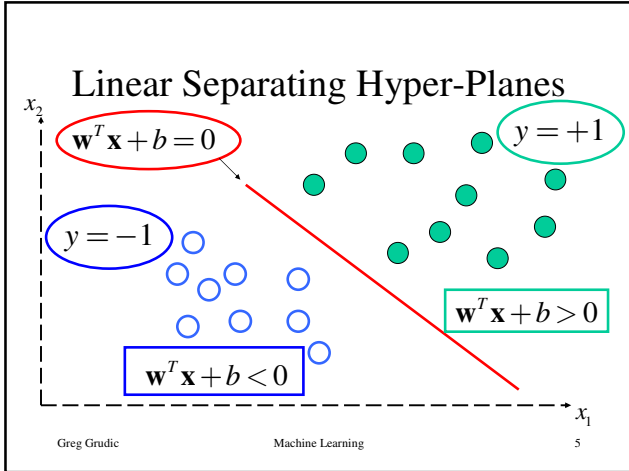
- Classification as a problem of finding optimal (canonical) linear hyperplanes.
- Optimal Linear Separating Hyperplanes:
  - In Input Space
  - In Kernel Space

## Linear Separating Hyper-Planes

How many lines can separate these points?

Which line should we use?





### Linear Separating Hyper-Planes

- Given data:  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$
- Finding a separating hyperplane can be posed as a constraint satisfaction problem
  - $\forall i \in (1, \dots, N)$ , find  $\mathbf{w}$  such that
    - $\mathbf{w}^T \mathbf{x}_i + b \geq +1$  if  $y_i = +1$
    - $\mathbf{w}^T \mathbf{x}_i + b \leq -1$  if  $y_i = -1$
- Or, equivalently
  - $y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 \geq 0, \quad \forall i$

Greg Grudic      Machine Learning      6

### The Margin of a Classifier

- Take any hyper-plane (P0) that separates the data
- Put a parallel hyper-plane (P1) on a point in class 1 closest to P0
- Put a **second** parallel hyper-plane (P2) on a point in class -1 closest to P0
- The margin is the perpendicular distance between P1 and P2*

Greg Grudic      Machine Learning      7

### Margin For Canonical Hyperplane

Calculating the Margin:

Note:

$$\langle \mathbf{w}, \mathbf{x}_1 \rangle + b = +1$$

$$\langle \mathbf{w}, \mathbf{x}_2 \rangle + b = -1$$

$$\Rightarrow \langle \mathbf{w}, (\mathbf{x}_1 - \mathbf{x}_2) \rangle = 2$$

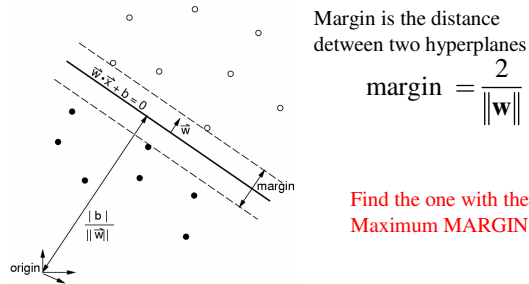
$$\Rightarrow \left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, (\mathbf{x}_1 - \mathbf{x}_2) \right\rangle = \frac{2}{\|\mathbf{w}\|}$$

$$\langle \mathbf{w}, \mathbf{x} \rangle = \sum_{i=1}^d w_i x_i$$

$$\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$$

Greg Grudic      Machine Learning      8

## How do SVMs choose the boundary?



Greg Grudic

Machine Learning

9

## SVM: Constraint Optimization Problem

- Given data:

$$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$$

- Minimize  $\|\mathbf{w}\|^2$  subject to:

$$y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 \geq 0, \quad \forall i = (1, \dots, N)$$

Greg Grudic

Machine Learning

10

## The Lagrange Function Formulation

Introduce Lagrange multipliers  $\alpha_i \geq 0$  and a Lagrangian

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^m \alpha_i (y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1).$$

$L$  has to be minimized w.r.t. the *primal variables*  $\mathbf{w}$  and  $b$  and maximized with respect to the *dual variables*  $\alpha_i$

- if a constraint is violated, then  $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 < 0 \rightarrow$ 
  - $\alpha_i$  will grow to increase  $L$  — how far?
  - $\mathbf{w}$ ,  $b$  want to decrease  $L$ ; i.e. they have to change such that the constraint is satisfied. If the problem is separable, this ensures that  $\alpha_i < \infty$ .
- similarly: if  $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 > 0$ , then  $\alpha_i = 0$ ; otherwise,  $L$  could be increased by decreasing  $\alpha_i$  (*KKT conditions*)

B. Schölkopf, Calvoerra, February 2002

Greg Grudic

Machine Learning

11

## Derivation of the Dual Problem

- At the saddle point (extremum)

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

- This gives the conditions

$$\sum_{i=1}^N \alpha_i y_i = 0, \quad \mathbf{w} = \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i$$

- Substitute into  $L(\mathbf{w}, b, \boldsymbol{\alpha})$  to get the dual problem

Greg Grudic

Machine Learning

12

## Dual Problem

- Maximize

$$W(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

- Subject to

$$\alpha_i \geq 0, \quad i = 1, \dots, N$$

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

Greg Grudic

Machine Learning

13

## Support Vector Expansion (1)

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

$$y_i [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] > 1 \Rightarrow \alpha_i = 0 \rightarrow \mathbf{x}_i \text{ irrelevant}$$

OR

$$y_i [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] = 1 \text{ (On Margin)} \quad \mathbf{x}_i \text{ Support Vector}$$

Greg Grudic

Machine Learning

14

## Support Vector Expansion (2)

$$f(\mathbf{x}) = \text{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$$

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

OR

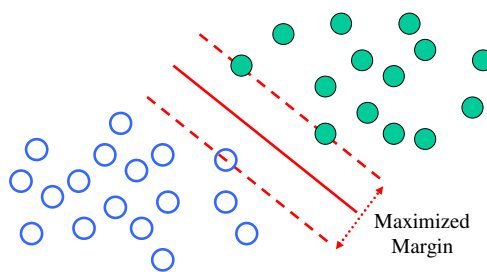
$$f(\mathbf{x}) = \text{sgn} \left( \sum_{i=1}^N \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b \right)$$

Greg Grudic

Machine Learning

15

## What are the Support Vectors?



Greg Grudic

Machine Learning

16

## Why do we want a model with only a few SVs?

- Leaving out an example that does not become an SV *gives the same solution!*
- Theorem (Vapnik and Chervonenkis, 1974):** Let #SV(N) be the number of SVs obtained by training on N examples randomly drawn for  $P(X, Y)$ , and E be an expectation. Then

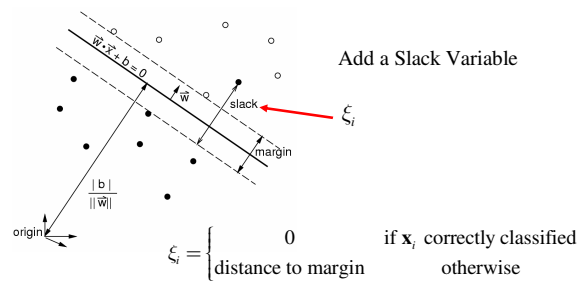
$$E[\text{Prob}(\text{test error})] \leq \frac{E[\text{\#SV}(N)]}{N}$$

Greg Grudic

Machine Learning

17

## What Happens When Data is Not Separable: **Soft Margin SVM**



Greg Grudic

Machine Learning

18

## Soft Margin SVM: Constraint Optimization Problem

- Given data:

$$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$$

- Minimize  $\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i$  subject to:

$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \forall i = (1, \dots, N)$$

Greg Grudic

Machine Learning

19

## Dual Problem (Non-separable data)

- Maximize

$$W(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

- Subject to

$$0 \leq \alpha_i \leq C, \quad i = 1, \dots, N$$

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

Greg Grudic

Machine Learning

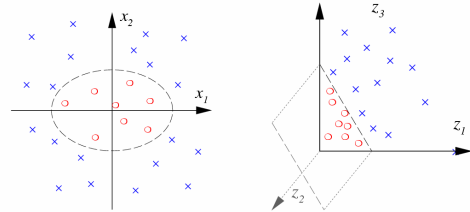
20

## Same Decision Boundary

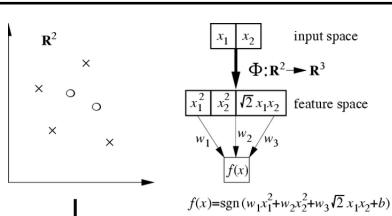
$$f(\mathbf{x}) = \text{sgn} \left( \sum_{i=1}^N \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b \right)$$

## Mapping into Nonlinear Space

$$\begin{aligned} \Phi: \mathbb{R}^2 &\rightarrow \mathbb{R}^3 \\ (x_1, x_2) &\mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2} x_1 x_2, x_2^2) \end{aligned}$$



## Nonlinear Data?



$$f(x) = \text{sgn}(w_1 x_1^2 + w_2 x_2^2 + w_3 \sqrt{2} x_1 x_2 + b)$$

## Nonlinear SVMs

- KEY IDEA:** Note that both the decision boundary and dual optimization formulation rely on dot products in input space only!

$$\langle \mathbf{x}_i, \mathbf{x} \rangle$$

$$f(\mathbf{x}) = \text{sgn} \left( \sum_{i=1}^N \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b \right)$$

$$W(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

## Kernel Trick

Replace  $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$

with

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$$

Can use the same algorithms in nonlinear kernel space!

## Nonlinear SVMs

Maximize:

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

Boundary:

$$f(\mathbf{x}) = \text{sgn} \left( \sum_{i=1}^N \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \right)$$

## Need Mercer Kernels

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_j) &= \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle \\ &= \langle \Phi(\mathbf{x}_j), \Phi(\mathbf{x}_i) \rangle \\ &= K(\mathbf{x}_j, \mathbf{x}_i) \end{aligned}$$

## Gram (Kernel) Matrix

**Training Data:**  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$

$$K = \begin{pmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & \dots & K(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ K(\mathbf{x}_N, \mathbf{x}_1) & \dots & K(\mathbf{x}_N, \mathbf{x}_N) \end{pmatrix}$$

Properties:

- Positive Definite Matrix
- Symmetric
- Positive on diagonal
- N by N

## Commonly Used Mercer Kernels

- Polynomial

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + c)^d$$

- Sigmoid

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\kappa \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \theta)$$

- Gaussian

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$$

## Soft Margin SVMs

*C-SVM* [15]: for  $C > 0$ , minimize

$$\tau(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \xi_i$$

subject to  $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$ ,  $\xi_i \geq 0$  (margin  $2/\|\mathbf{w}\|$ )

*$\nu$ -SVM* [55]: for  $0 \leq \nu < 1$ , minimize

$$\tau(\mathbf{w}, \boldsymbol{\xi}, \rho) = \frac{1}{2} \|\mathbf{w}\|^2 - \nu \rho + \frac{1}{m} \sum_i \xi_i$$

subject to  $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq \rho - \xi_i$ ,  $\xi_i \geq 0$  (margin  $2\rho/\|\mathbf{w}\|$ )

## Duals, Using Kernels

*C-SVM* dual: maximize

$$W(\boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

subject to  $0 \leq \alpha_i \leq C$ ,  $\sum_i \alpha_i y_i = 0$ .

*$\nu$ -SVM* dual: maximize

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

subject to  $0 \leq \alpha_i \leq \frac{1}{m}$ ,  $\sum_i \alpha_i y_i = 0$ ,  $\sum_i \alpha_i \geq \nu$

In both cases: *decision function*:

$$f(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^m \alpha_i y_i k(\mathbf{x}, \mathbf{x}_i) + b\right)$$

## SVM Training

- naive approach: the complexity of maximizing

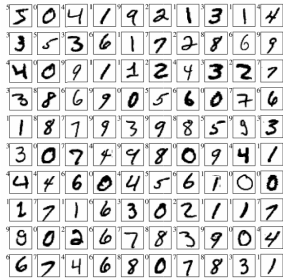
$$W(\boldsymbol{\alpha}) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

scales with the third power of the training set size  $m$

- only SVs are relevant  $\rightarrow$  only compute  $(k(\mathbf{x}_i, \mathbf{x}_j))_{ij}$  for SVs. Extract them iteratively by cycling through the training set in chunks [63].
- in fact, one can use chunks which do not even contain all SVs [42]. Maximize over these sub-problems, using your favorite optimizer.
- the extreme case: by making the sub-problems very small (just two points), one can solve them analytically [45].

## MNIST: A SVM Success Story

- Handwritten character benchmark
  - 60,000 training and 10,000 testing
  - Dimension  $d = 28 \times 28$



## Results on Test Data

Classifier	test error
linear classifier	8.4%
3-nearest-neighbour	2.4%
SVM	1.4%
Tangent distance	1.1%
LeNet4	1.1%
Boosted LeNet4	0.7%
Translation invariant SVM	0.56%

SVM used a polynomial kernel of degree 9.

## SVM (Kernel) Model Structure

