IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

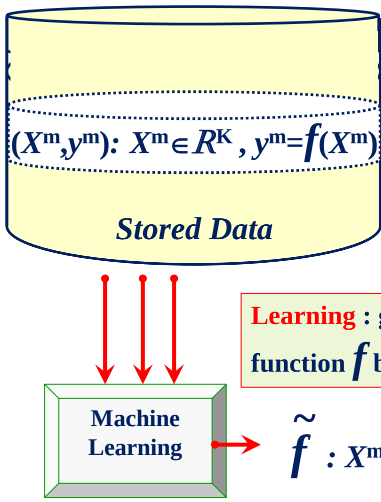
Machine Learning Approaches - SVM

B. Solaiman
Basel.solaiman@imt-atlantique.fr
Départ. Image &Traitement de l'Information
Brest, France

2

Introduction

Introduction



A Pattern (sample, instance, example, ….) is represented by a set of K features, or attributes, viewed as a K-dimensional feature vector $X \in R^K$

The pattern "may" be associated with a target variable $y = f(X)$ (in case of supervised learning)

f : Unknown function associating X and y

Learning : given the *training set*, estimate the prediction function f by minimizing "a" prediction error

$$\tilde{f} : X^m \in R^K , y^m = \tilde{f}(X^m)$$

Introduction

y

No Target Variable y	$\Rightarrow$	*Unlabeled Training Data*	Data Structure Discovery
$y \in L = \{C_1, C_2, \ldots Cn\}$ (*L: set of discrete labels*)	$\Rightarrow$	*Labeled Training Data*	Classification
$y \in R$	$\Rightarrow$	*Labeled Training Data*	Regression

Regression problem

Instead of predicting the class of an input pattern, we want to predict continuous values

Introduction

Natures of a feature

- **Numerical, or quantitative, Feature**
- **Symbolic, or qualitative, Feature**
  - **Nominal Feature:**
    Feature values are symbols (= and ≠ are the only operations we can conduct)
  - **Ordinal Feature:**
    Feature values are "ordered" symbols (=, ≠, <, >, ≤ and ≥ are the only operations we can conduct)

Introduction

Distance between patterns

Numerical Features

$X = [x_1, x_2, .., x_n]$ & $Y = [y_1, y_2, .., y_n]$ $x_i, y_i \in \Re$

Minkowski Distance

$$d_p(X,Y) = \left\{ \sum_{i=1}^{n} |x_i - y_i|^p \right\}^{\frac{1}{p}}$$

Manhattan distance : $d_1(X,Y) = \sum_{i=1}^{n} |x_i - y_i|$ $(P = 1)$

Euclidian distance : $d_2(X,Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$ $(P = 2)$

Max distance : $d_\infty(X,Y) = \max_{i=1}^{n} |x_i - y_i|$ $(P = \infty)$

Introduction

7

Distance between patterns

Nominal features

$$X = [x_1, x_2, .., x_n] \;\; \& \;\; Y = [y_1, y_2, .., y_n]$$

$$x_i, y_i \in \Omega \subseteq \Omega_1 \times \Omega_2 \times \ldots \times \Omega_n$$

$$dist(X, Y) = \frac{n - Q}{n}$$

Q: Number of features having the same modality in *X* and *Y*

Introduction

8

Distance between patterns

Nominal binary features

$$X = [x_1, x_2, .., x_n] \;\; \& \;\; Y = [y_1, y_2, .., y_n] \;\; x_i, y_i \in \Omega \subseteq \{0, 1\}$$

Confusion Matrix

$$\begin{array}{cc|cc} & & \multicolumn{2}{c}{Y} \\ & & 1 & 0 \\ \hline X & 1 & a & b \\ & 0 & c & d \end{array}$$

a : nb of feature assuming « 1 » in X and « 1 » in Y
b : nb of feature assuming « 1 » in X and « 0 » in Y
c : nb of feature assuming « 0 » in X and « 1 » in Y
d : nb of feature assuming « 0 » in X and « 0 » in Y

Introduction

Distance between patterns

Nominal binary features

Case 1: Binary Symmetric Features (0 and 1 have the same importance)
Simple Matching Coefficient

$$dist(X, Y) = \frac{b + c}{a + b + c + d}$$

		Y = 1	Y = 0
X	1	a	b
	0	c	d

Example

X	1	1	1	0	1	0	0
Y	0	1	1	0	0	1	0

$$dist(X, Y) = \frac{2 + 1}{2 + 2 + 1 + 2} = 3 / 7 = 0.429$$

Introduction

Distance between patterns

Nominal binary features

Case 2: Binary Non Symmetric Features
(1 is more important than 0)
Jaccard' *Coefficient*

$$dist(X, Y) = \frac{b + c}{a + b + c}$$

		Y = 1	Y = 0
X	1	a	b
	0	c	d

Exemple

X	1	1	1	0	1	0	0
Y	0	1	1	0	0	1	0

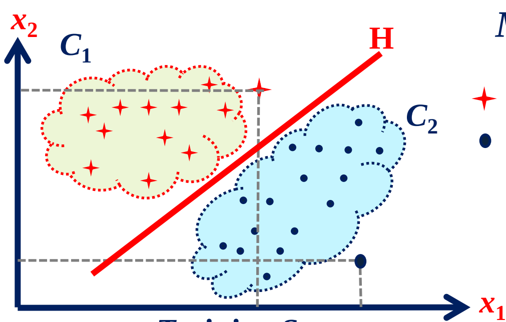
$$dist(X, Y) = \frac{2 + 1}{2 + 2 + 1} = 3 / 5$$

Support Vector Machines (SVM)

Support Vector Machines (SVM)

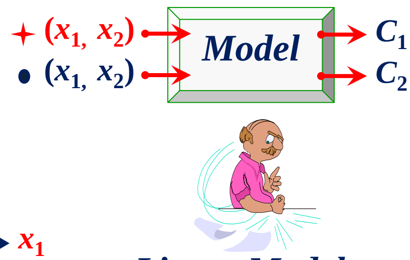
- *Linear Models for Classification*
- *Perceptron*
- *A Bit of Geometry*
- *Hard (Linear) Support Vector Machines* (H-*SVM*)
- *Soft Margin Support Vector Machines* (SM-*SVM*)
- *Kernel-based Support Vector Machines*

Linear Models for Classification

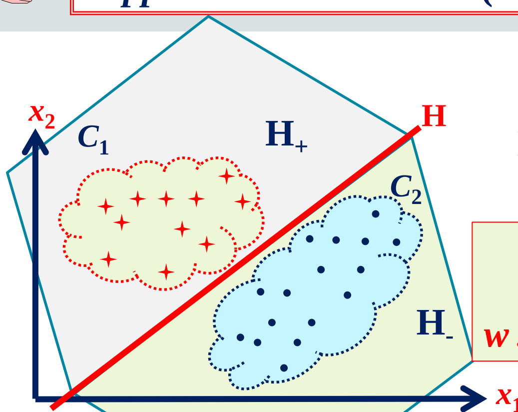


Training Set
(Binary classification
=
Two classes)

Model for Classification ?



Linear Model :
Hyperplane
Separation Surface H



How does it work ?

Hyperplane equation :

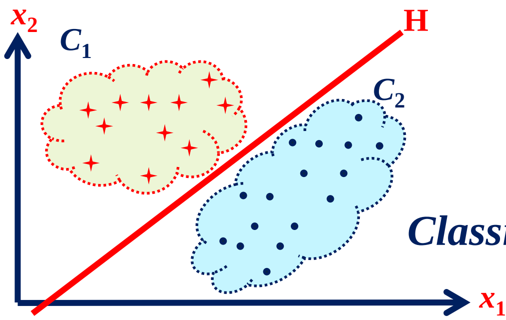
$$w \,.\, X + b = 0$$

$X = (x_1, x_2, \ldots, x_n)$
$w = (\omega_1, \omega_2, \ldots, \omega_n)$
$w \,.\, X = x_1.\omega_1 + x_2.\omega_2 + \ldots + x_n.\omega_n$

IF : X "on" H: $w \,.\, X + b = 0$
IF : $X \in H_+$: $w \,.\, X + b > 0$
IF : $X \in H_-$: $w \,.\, X + b < 0$

Support Vector Machines (SVM)

15



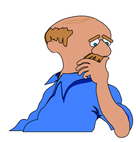
Linear Model Classification Decision Rule :

X → **Computation of:** $w \,.\, X + \mathrm{b}$ → If $w \,.\, X + \mathrm{b} > 0$ Then Decide C_1; If $w \,.\, X + \mathrm{b} > 0$ Then Decide C_2 →

Support Vector Machines (SVM)

16

Linear Model for Binary Classification



Input Training dataset

$X = (x_1, x_2,\ldots, x_n), y$

→ *Find:* $w = (\omega_1, \omega_2,\ldots, \omega_n)$, b

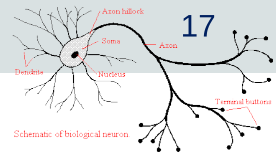
$w \,.\, X + \mathrm{b} > 0$ if $y = +1$

$w \,.\, X + \mathrm{b} < 0$ if $y = -1$

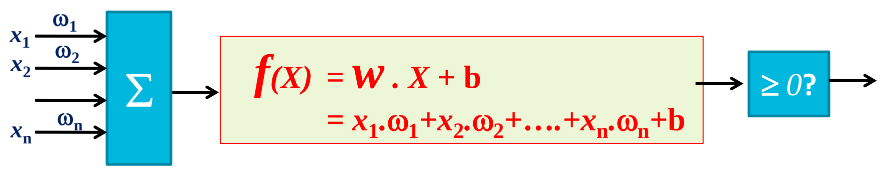
→ $w \,.\, X + \mathrm{b} = 0$

Support Vector Machines (SVM)

17



Perceptron, Rosenblatt - 1958



$$b = \omega_0 \,,\, x_0 = 1 \longrightarrow f(X) = \mathbf{w} \,.\, X \quad \text{(dot product)}$$

Decision Rule:

If $f(X) = \mathbf{w} \,.\, X > 0$ *Then* $+1$

If $f(X) = \mathbf{w} \,.\, X < 0$ *Then* -1

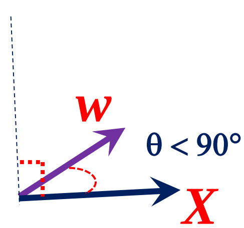
Support Vector Machines (SVM)

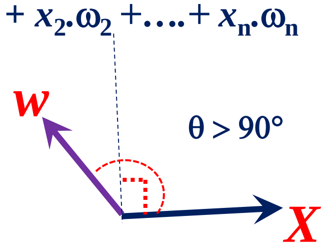
18

A Bit of Geometry

Dot Product $< w, X >$

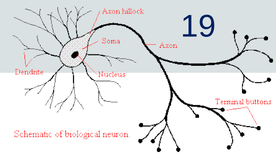
$$= \mathbf{w} \,.\, X$$
$$= \|\mathbf{w}\| \, \|X\| \cos\theta$$
$$= x_1.\omega_1 + x_2.\omega_2 + \ldots. + x_n.\omega_n$$





$w.\, X > 0$ $\qquad$ $w.\, X < 0$

Support Vector Machines (SVM)



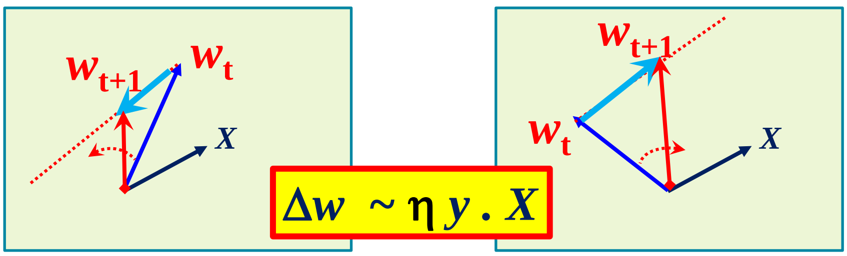
Perceptron, Rosenblatt - 1958

Decision Rule: *If* $f(X) = \mathbf{w} \,.\, X > 0$ *Then* +1

If $f(X) = \mathbf{w} \,.\, X < 0$ *Then* -1

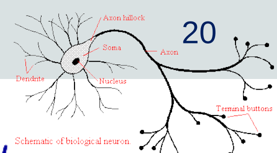
Types of error:

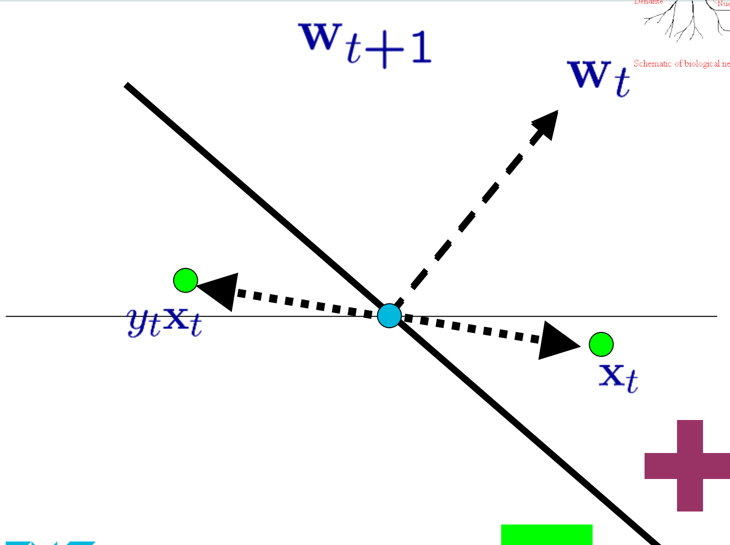
$f(X) = \mathbf{w}.\, X > 0$ *whereas* $y = -1$ $f(X) = \mathbf{w} \,.\, X < 0$ *whereas* $y = +1$



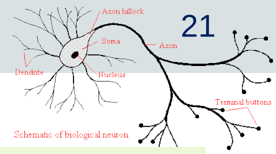
$$\Delta w \sim \eta\, y \,.\, X$$

Support Vector Machines (SVM)





Support Vector Machines (SVM)



21

Perceptron, Rosenblatt - 1958

Perceptron Algorithm :

Random initialization of $(\omega_1, \omega_2, \ldots, \omega_n)$, **b**
Pick training samples X **one by one**
Predict class of X **using current** $(\omega_1, \omega_2, \ldots, \omega_n)$, **b**

$$y' = Sign(W \cdot X)$$

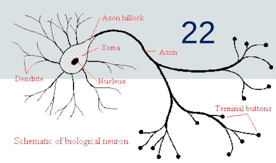
If y' **is correct (i.e.,** $y = y'$**) :** **No change:** $W_{t+1} = W_t$

If y' **is wrong: Adjust** W_t **:** $W_{t+1} = W_t + \eta \cdot y_t \cdot X_t$

- η is the learning rate parameter
- X_t is the t-th training example
- y_t is true t[th] class label ({+1, -1})

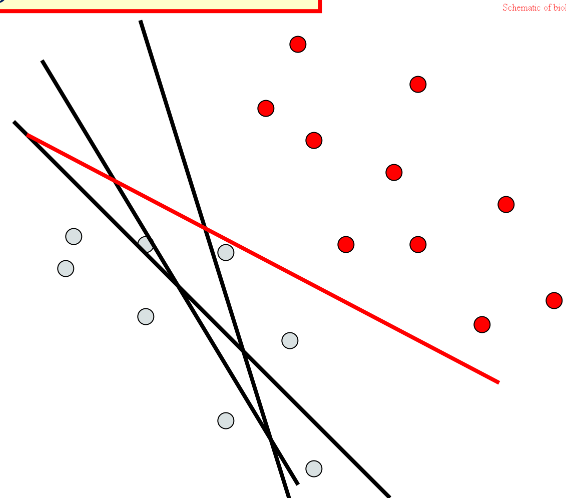
Machine Learning Approaches ----------- B. Solaiman

Support Vector Machines (SVM)



22

Perceptron, Rosenblatt - 1958



Machine Learning Approaches ----------- B. Solaiman

Support Vector Machines (SVM)

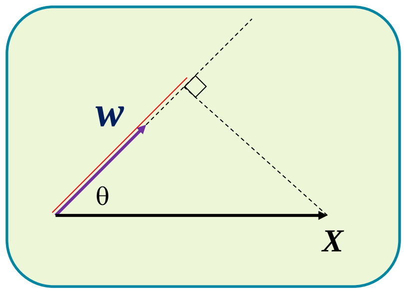
23

A Bit of Geometry

Projection of X onto w

$$= \|X\| \; \text{Cos}\theta$$

$$= w \,.\, X \;/\; \|w\|$$



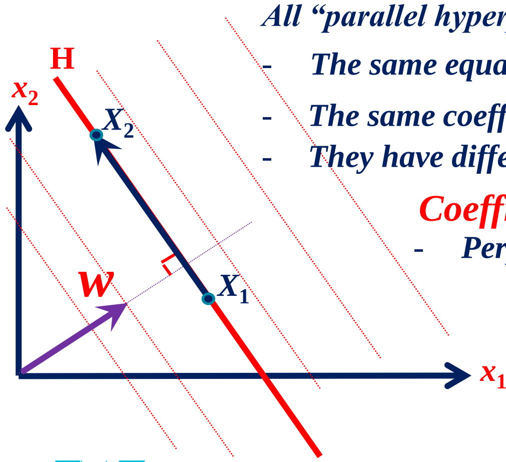
Machine Learning Approaches ----------- B. Solaiman

Support Vector Machines (SVM)

24

A Bit of Geometry

Parallel Hyperplanes



All "parallel hyperplanes" have:

- *The same equation $w \,.\, X + b = 0$;*
- *The same coefficients vector w;*
- *They have different origin translation values b*

Coefficients vector : W

- *Perpendicular to H ?*

$$w \,.\, (X_1X_2) \;=\; w \,.\, (X_2 - X_1)$$
$$= w.X_2 - w.X_1$$
$$= -b - (-b) = 0$$

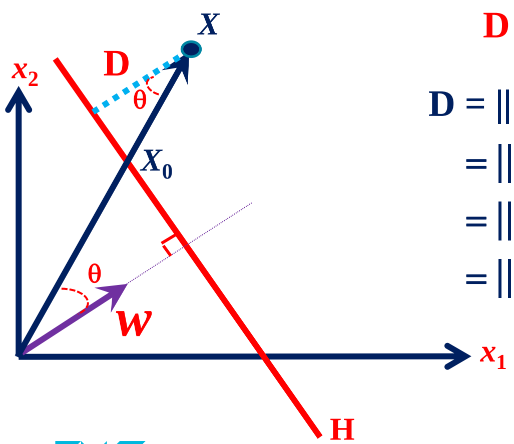
Machine Learning Approaches ----------- B. Solaiman

Support Vector Machines (SVM)

25

A Bit of Geometry

Distance between an instance X and H



$$D = \mathrm{dist}(X, H) = ??$$

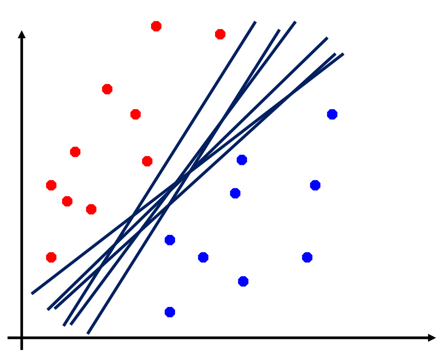
$$\begin{aligned} D &= \| X_0X \| \cos\theta \\ &= \| w.\, X_0X \| \;/\; \| w \| \\ &= \| w \,.\, X - w \,.\, X_0 \| \;/\; \| w \| \\ &= \| w \,.\, X + b \| \;/\; \| w \| \end{aligned}$$

Support Vector Machines (SVM)

26

Hard (Linear) Support Vector Machines (H-SVM)

Which of the linear separators is optimal?

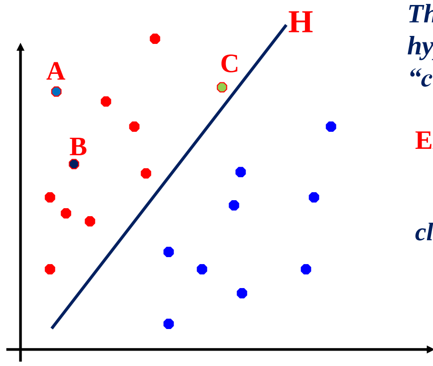


Support Vector Machines (SVM)

27

Hard (Linear) Support Vector Machines (H-SVM)

Largest Margin Concept:



The distance from the separating hyperplane, H, corresponds to the "confidence" of decision

Example:

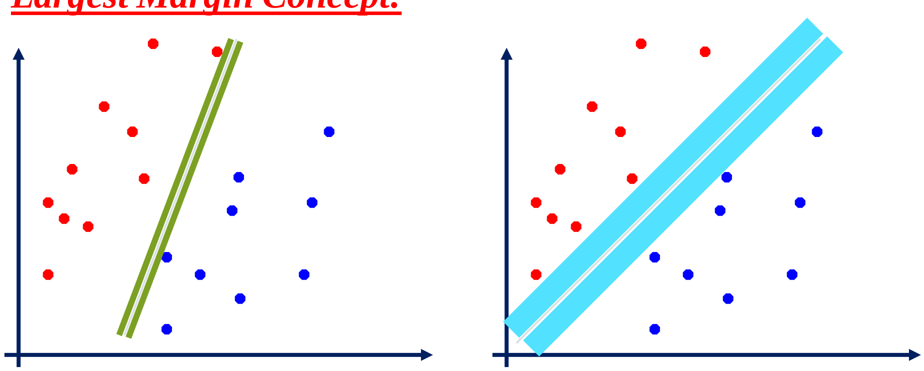
We are more sure and confident about the class of **A** *and* **B** *than of* **C**

Support Vector Machines (SVM)

28

Hard (Linear) Support Vector Machines (H-SVM)

Largest Margin Concept:

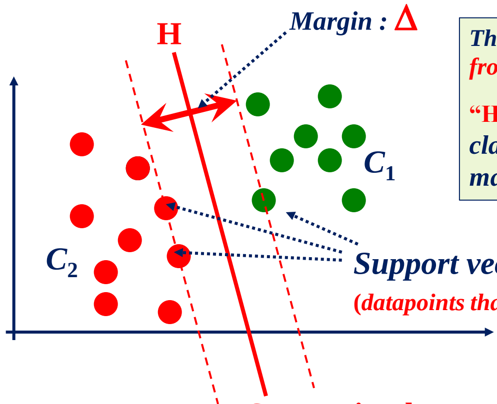


The **margin** *of a linear classifier is the width that the boundary could be increased by before hitting a datapoint instance*

Support Vector Machines (SVM)

Hard (Linear) Support Vector Machines (H-SVM)

Largest Margin Concept :



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

"H-SVM" : The maximum margin linear classifier is the linear classifier with the maximum margin

Support vectors

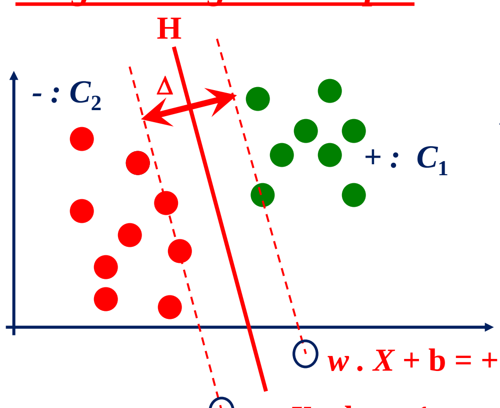
(datapoints that the margin pushes up against)

Separating hyperplane : $w \cdot X + b = 0$

Support Vector Machines (SVM)

Hard (Linear) Support Vector Machines (H-SVM)

Largest Margin Concept :

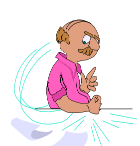


$w \cdot X + b = +1$

$w \cdot X + b = -1$

Margin width : $\Delta = ?$

$$\Delta = 2 \cdot |w \cdot X + b| \;/\; \|w\|$$



$$\boxed{\Delta = 2 \;/\; \|w\|}$$

Support Vector Machines (SVM)

Hard (Linear) Support Vector Machines (H-SVM)

Vapnik, 1965; Vapnik, 1995 :

$\{(X^m, y^m),\ m = 1,..., M,\ X^m \in \mathcal{R}^n,\ y^m \in \{+1, -1\}\}$

H-SVM formulation : Find w & b such that

Maximize :

$$\Delta = 2\ /\ \|\, w \,\|$$

Minimize:

$$\|\, w \,\|^2 / 2$$

Subject to **Support vectors** *constraints :*

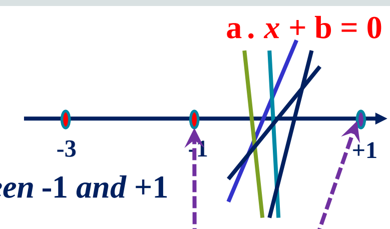
$$y^m . (w . X^m + b) \geq +1 \quad \textit{for all } m$$

Quadratic Optimization problem, subject to linear constraints

Support Vector Machines (SVM)

1D Example

Dataset : $\{(-3,-1), (-1,-1), (+2,+1)\}$



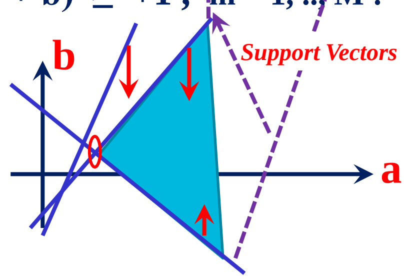
H : *All lines with a cross point between -1 and +1*

Margin constraints: $y^m . (w . X^m + b) \geq +1$, m = 1, .., M :

a . (-3) + b < -1 → b < 3a - 1

a . (-1) + b < -1 → b < a - 1

a . (+2) + b > +1 → b > - 2a + 1



Minimize

$\|\, w \,\|^2 = \|a\|^2$ → a = 0.66

→ b = 0.33

Hard (Linear) Support Vector Machines (H-SVM)

RAPPEL: QUADRATIC PROGRAMMING PROBLEM

$$F(w\,,b) = b + \sum a_i\,\omega_i + \sum\sum q_{i,j}\,\omega_i\,\omega_j \qquad w = (\omega_1, \omega_2, \ldots\omega_n)\ , b$$

Quadratic objective Function $F(w\,,b)$ → **Quadratic Programming Numerical Analysis** → $w=(\omega_1, \omega_2, \ldots\omega_n)\ , b$

M *Linear Constraints*

$$\sum c_{ij}\,\omega_i \le d_j \qquad j = 1, 2, \ldots, M$$

Hard (Linear) Support Vector Machines (H-SVM)

Optimization using Lagrange Multipliers

Lagrangian :

$$L = \|\,w\,\|^2 / 2 \ + \sum_{m=1,..,M} \lambda_m \left\{ 1 - \left[y^m \,.\, (w \,.\, X^m + b) \right] \right\}$$

Lagrange Multipliers

The constraints are added into the optimization by adding extra variables (called Lagrange multipliers)

$$\|\,w\,\|^2 = (\omega_1)^2 + (\omega_2)^2 + \ldots + (\omega_n)^2$$

Support Vector Machines (SVM)

Hard (Linear) Support Vector Machines (H-SVM)

Optimization using Lagrange Multipliers

Objective Function:

$$L = \| w \|^2 / 2 + \sum_{m=1,..,M} \lambda_m \{ 1 - [y^m . (w . X^m + b)] \}$$

Linear sum of samples

Setting the gradient of L w.r.t. w and b to zero, we hav

$$\frac{\partial L}{\partial w} = w - \sum_{m=1,..,M} \lambda_m y^m . X^m = 0$$

$$\boxed{w^* = \sum_{m=1,..,M} \lambda_m \; y^m . X^m}$$

$$\frac{\partial L}{\partial b} = - \sum_{m=1,..,M} \lambda_m y^m = 0$$

$$\sum_{m=1,..,M} \lambda_m \; y^m = 0$$

Support Vector Machines (SVM)

$$L = \| w \|^2 / 2 + \sum_{m=1,..,M} \lambda_m \{ 1 - [y^m . (w . X^m + b)] \}$$

$$w^* = \sum_{m=1,..,M} \lambda_m \; y^m . X^m$$

$$L = (1/2) . [\sum_{m=1,..,M} \lambda_m y^m . X^m] . [\sum_{m=1,..,M} \lambda_m y^m . X^m] - [\sum_{m=1,..,M} \lambda_m y^m . X^m \{ \sum_{k=1,..,M} \lambda_k y^k . X^k \}] - \sum_{m=1,..,M} \lambda_m y^m . b + \sum_{m=1,..,M} \lambda_m$$

The optimization problem becomes : (called : Dual Problem)

Find λ_m such that:

$$\boxed{L = - (1/2) . \sum_{m=1,..,M} \sum_{k=1,..,M} \lambda_m \lambda_k \; y^m y^k . [X^m . X^k] + \sum_{m=1,..,M} \lambda_m}$$

is maximized

where

$$\boxed{\sum_{m=1,..,M} \lambda_m \; y^m = 0}$$

$$\boxed{\lambda_m \geq 0, \; m = 1,.., M}$$

Support Vector Machines (SVM)

37

$$L = -(1/2) \,.\, \sum_{m=1,..,M} \sum_{k=1,..,M} \lambda_m \lambda_k \; y^m y^k .\, [X^m . X^k] + \sum_{m=1,..,M} \lambda_m$$

- *Quadratic optimization problems are a well-known mathematical programming problems for which several algorithms exist;*

- *Once resolved, most of λ_m are 0, only a small number have $\lambda_m > 0$; the corresponding X^m s are the support vectors*

Machine Learning Approaches ----------- B. Solaiman

Support Vector Machines (SVM)

38

Given a solution $\lambda_1 \dots \lambda_M$ to the dual problem, solution to the primal (i.e., coefficients vector w^* and origin shift factor b^* solution) is given by :

Learned weight

$$w^* = \sum_{m \in \text{Support Vectors}} \lambda_m \, y^m \,.\, X^m$$

Support Vectors

$$b^*(k) = y^k - \sum_{m=1,\dots,M} \lambda_m \, y^m \,.\, X^m X^k$$

For any k such that $\lambda_k > 0$

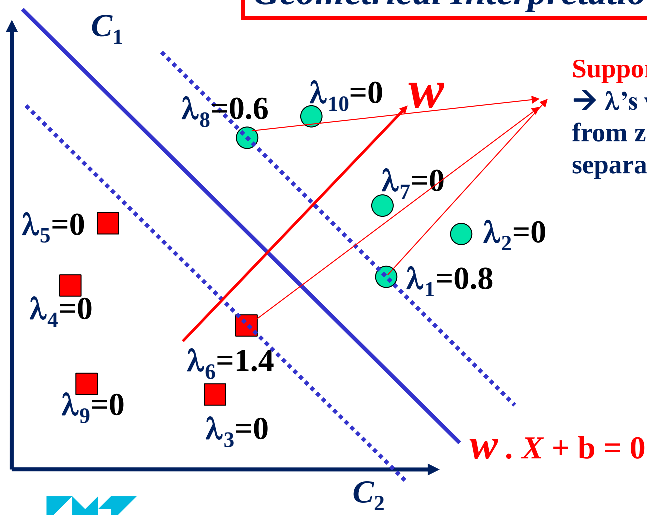
$$b^* = (1/K) \sum_k b^*(k) \quad \text{(K denotes the nb of SVs)}$$

Machine Learning Approaches ----------- B. Solaiman

Support Vector Machines (SVM)

39

Geometrical Interpretation



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

Support Vector Machines (SVM)

40

Decision rule : Given a new data instance X

Then, the classifying function is
(*note that we don't need w* explicitly*):

$$f(X) = \sum_{m \in \text{Support Vectors}} \lambda_m \, y^m \, . \, X^m \, . \, X \; + \; b^*$$

If $f(X) > 0$ *Then* *decide* C_1;
Else *decide* C_2

Support Vector Machines (SVM)

- *Notice that* $f(X)$ ***relies on the dot product*** *between the unknown instance X and the support vectors* X^m

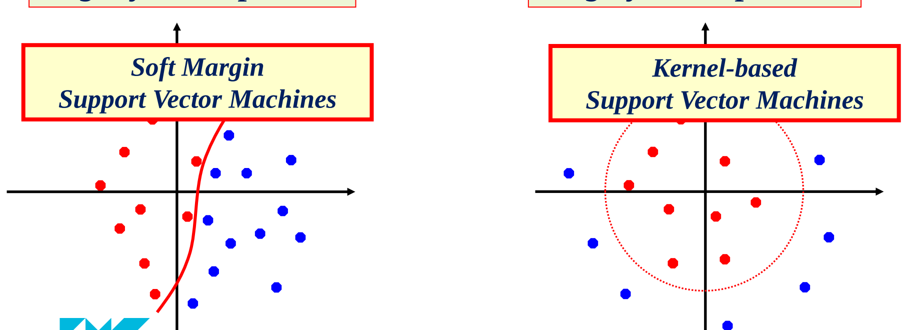
- *Also keep in mind that solving the optimization problem involved computing the dot products* $X^m X^k$ *between all training data instances*

Support Vector Machines (SVM)

Soft Margin Support Vector Machines (SM-SVM)

Non Linear Separability

Slightly non separable | *Highly non separable*



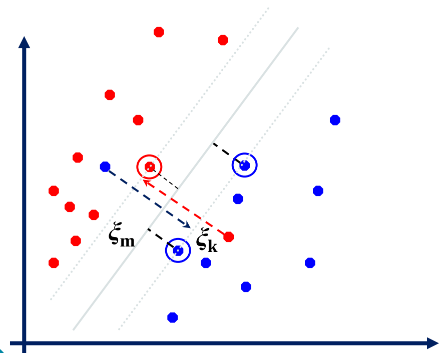
Support Vector Machines (SVM)

Soft Margin Support Vector Machines (SM-SVM)

To allow errors in data, we relax the margin constraints by introducing slack variables, ξ_m (≥ 0) :

$$w.X^m + b \geq +1-\xi_m\,, \quad \text{if } y^m = +1$$

$$w.X^m + b \leq -1+\xi_m\,, \quad \text{if } y^m = -1$$



The new constraints:

$$y^m.(w.X^m+b) \geq 1-\xi_m, \; m=1, \ldots, M$$

$$\xi_m \geq 0\,, \; m=1, \ldots, M$$

ξ_m: *Measure of margin violation by an erroneous instance* X^m

Support Vector Machines (SVM)

Assigning an "extra cost" for errors in the objective function that we are going to optimize:

Minimize:

$$L = \|w\|^2/2 + C \sum_{m=1,\ldots,M} \xi_m$$

Subject to:

$$y^m.(w.X^m + b) \geq 1 - \xi_m, \; m=1, \ldots, M$$

$$\xi_m \geq 0\,, \; m=1, \ldots, M$$

$\sum_{m=1,\ldots,M} \xi_m$: *Total margin violation*

C : **Large C means a higher penalty to errors**

Support Vector Machines (SVM)

Optimization using Lagrange Multipliers

$$L = \|w\|^2/2 + C\sum_{m=1,..,M} \xi_m - \sum_{m=1,..,M} \lambda_m \{y^m.(w.X^m + b) - 1 + \xi_m\} - \sum_{m=1,..,M} \mu_m \xi_m$$

Minimizing ↔ Maximizing the margin

Total margin violation error

$\lambda_m, \mu_m \geq 0$
Lagrange Multipliers

Objective function
to be minimized w.r.t. w, b & ξ_m

Tradeoff (i.e., relative importance) between error and margin

Support Vector Machines (SVM)

$$L = \|w\|^2/2 + C\sum_{m=1,..,M} \xi_m - \sum_{m=1,..,M} \lambda_m \{y^m.(w.X^m + b) - 1 + \xi_m\} - \sum_{m=1,..,M} \mu_m \xi_m$$

Setting the gradient of L w.r.t. w, b and ξ_m to zero, we have :

$$\left.\begin{aligned} \frac{\partial L}{\partial w} &= w - \sum_{m=1,..,M} \lambda_m y^m . X^m = 0 \\ \frac{\partial L}{\partial b} &= - \sum_{m=1,..,M} \lambda_m y^m = 0 \end{aligned}\right\}$$

Same solution as for the H-SVM

$$\frac{\partial L}{\partial \xi_m} = C - \lambda_m - \mu_m = 0$$

$\mu_m \geq 0$
$\lambda_m \geq 0$
⇒ $\lambda_m \leq C$
(Otherwise, $C - \lambda_m \leq 0$, thus no μ_m will verify: $C - \lambda_m - \mu_m = 0$

Support Vector Machines (SVM)

$$w^* = \sum_{m=1,..,M} \lambda_m \, y^m \, . \, X^m$$

$$\sum_{m=1,..,M} \lambda_m \, y^m = 0$$

with the only difference w.r.t (H-*SVM*): $0 \leq \lambda_m \leq C$

Decision rule : Given a new data instance X

Then, the classifying function is

$$f(X) = \sum_{m \in \text{Support Vectors}} \lambda_m \, y^m \, . \, X^m \, . \, X + b^*$$

If $f(X) > 0$ *Then decide* C_1;

Else decide C_2

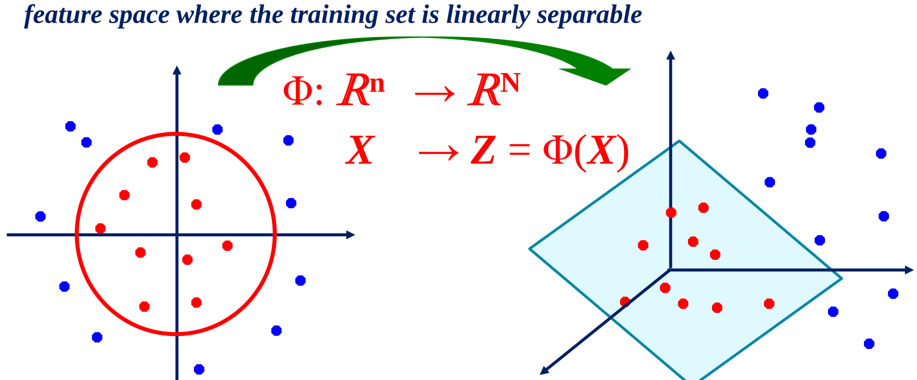
Support Vector Machines (SVM)

Kernel-based Support Vector Machines

General idea: *the original feature space is mapped to some higher-dimensional feature space where the training set is linearly separable*

Φ: $R^n \rightarrow R^N$

$X \rightarrow Z = \Phi(X)$

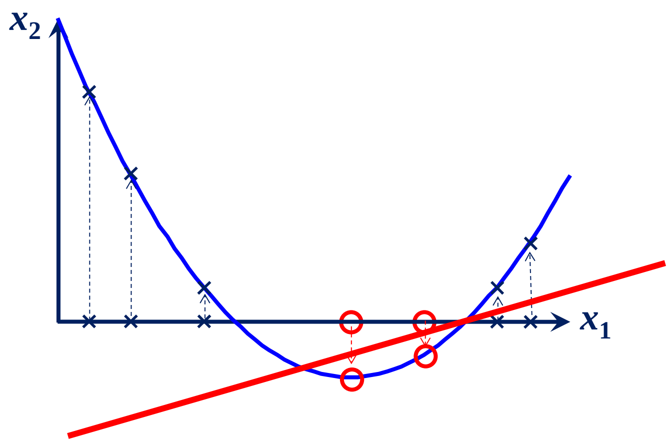


$X \in R^n$ $\qquad$ $Z \in R^N$, N > n

Support Vector Machines (SVM)

49

Kernel-based Support Vector Machines



Support Vector Machines (SVM)

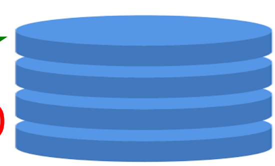
50

Kernel-based Support Vector Machines

R^n R^N

$\Phi: R^n \rightarrow R^N$

$X \rightarrow Z = \Phi(X)$



Nonlinearly separable dataset
$\{(X^m, y^m), X^m \in R^n, y^m \in \{+1, -1\}\}$

Linearly separable dataset
$\{(Z^m = \Phi(X^m), y^m), Z^m \in R^N, y^m \in \{+1, -1\}\}$

Do we need to know the explicit expression of $Z = \Phi(X)$?

SVM Machinery

Support Vector Machines (SVM)

51

The Kernel trick

Lagrangian to maximize

$$L = -(1/2) \cdot \sum_{m=1,..,M} \sum_{k=1,..,M} \lambda_m \lambda_k \, y^m y^k \cdot [Z^m \cdot Z^k] + \sum_{m=1,..,M} \lambda_m$$

Constraints

$$\sum_{m=1,..,M} \lambda_m \, y^m = 0 \qquad \lambda_m \geq 0, \; m = 1,.., M$$

Decision Rule

$$f(X) = \sum_{m \in \text{Support Vectors}} \lambda_m \, y^m \, . Z^m . Z + b^*$$

Bias term

$$b^* = y^k - \sum_{m=1,\ldots,M} \lambda_m \, y^m \, Z^m Z^k$$

We only deal with Z as far as Z-dot product is concerned

Support Vector Machines (SVM)

52

The Kernel trick

SVM Machinery (Quadratic Programming, Decision rule) needs

Z - Transformation

Z-dot product

X^m

$X^{m'}$

$$K(X^m, X^{m'}) = Z^m \cdot Z^{m'}$$

without explicit computation of $\Phi(X^m)$ and $\Phi(X^{m'})$

→ *Z-dot product*

$K(.,.)$ **:** *is called the Transformation Kernel*

Support Vector Machines (SVM)

53

Example 1

$$X = (x_1, x_2) \quad \ldots\ldots\ldots\ldots\ldots \; : \; X \in \mathbb{R}^2$$

$$\Phi(X) = (1, x_1, x_2, (x_1)^2, (x_2)^2, x_1 . x_2) :$$

Full 2^d order polynomial transformation

$$K(X, X') = Z . Z'$$

$$= 1 + x_1 x'_1 + x_2 x'_2 + (x_1)^2 (x'_1)^2 + (x_2)^2 (x'_2)^2 + x_1 . x'_1 . x_2 x'_2$$

Machine Learning Approaches ----------- B. Solaiman

Support Vector Machines (SVM)

54

Example 2

$$X = (x_1, x_2) \quad \ldots\ldots\ldots\ldots\ldots \; : \; X \in \mathbb{R}^2$$

Computing the Kernel without transforming X and X' ?

$$K(X, X') = (1 + X . X')^2$$

$$K(X, X') = (1 + X . X')^2 = (1 + x_1 x'_1 + x_2 x'_2)^2$$

$$= 1 + (x_1)^2 (x'_1)^2 + (x_2)^2 (x'_2)^2 + 2 x_1 x'_1 + 2 x_2 x'_2 + 2 x_1 x'_1 x_2 x'_2$$

This is the inner product of $Z = \Phi(X)$ and $Z' = \Phi(X')$ where

$$\Phi(X) = [1, (x_1)^2, (x_2)^2, \sqrt{2}\, x_1, \sqrt{2}\, x_2, \sqrt{2}\, x_1 x_2]$$

$$\Phi(X') = [1, (x'_1)^2, (x'_2)^2, \sqrt{2}\, x'_1, \sqrt{2}\, x'_2, \sqrt{2}\, x'_1 x'_2]$$

Machine Learning Approaches ----------- B. Solaiman

Support Vector Machines (SVM) 55

Example 2

Think about the computation complexity if :

- $X = (x_1, x_2, \ldots, x_d)$ ……… : $X \in \mathbb{R}^d$, d >> 2 and
- $K(X, X') = (1 + X \cdot X')^Q$, Q >> 2

Support Vector Machines (SVM) 56

Example 3 (Radial Basis Function Kernel)

Show that $K(X, X') = e^{-\alpha \|X - X'\|^2}$ *corresponds to an "inner product" in an infinite dimensional Z-Space (i.e., transforming X instances)*

Hint

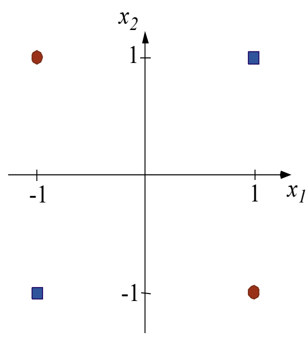
For d = 1 (X = x: i.e., scalar), α = 1:

- *Expand* $e^{-\alpha \|X - X'\|^2}$
- *Expand cross exponential term using Taylor series*

Support Vector Machines (SVM)

57

Example 4 XOR



Index i	X	y
1	(1,1)	1
2	(1,-1)	-1
3	(-1,-1)	1
4	(-1,1)	-1

$$K(X, X') = (1 + X \cdot X')^2$$

Machine Learning Approaches ----------- B. Solaiman

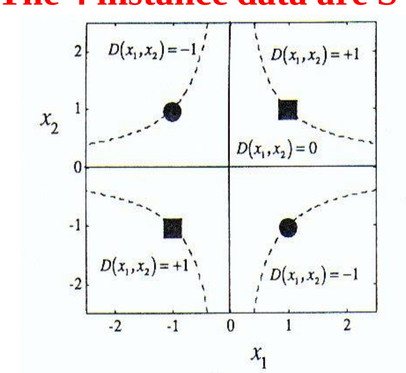
Support Vector Machines (SVM)

58

Example 4 XOR

Optimal Lagrange multipliers: $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 1/8$

The 4 instance data are SV



Machine Learning Approaches ----------- B. Solaiman

Support Vector Machines (SVM) 59

Example 5 Face Detection application

Training images database:

266 pictures: 150 faces + 116 non-faces

Preprocessing

- **Gray scale transformation**
- **Histogram equalization**
- **Adjust resolution to 30x40 pixel**

Training the SVM

based on the 266 training instances, a polynomial kernel

Support Vector Machines (SVM) 60

Example 5 Face Detection application

Test phase:

Given an input image:

- Moving 30x40 pixel sub window over the input image
- Histogram equalization of a sub window
- Classification by SVM
- Removing intersections

Support Vector Machines (SVM)

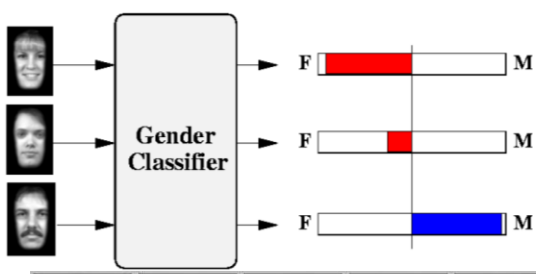
61

How many faces ?

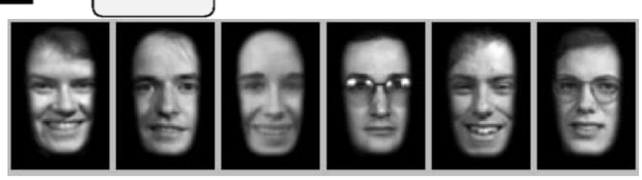
Support Vector Machines (SVM)

62

Example 6 Learning gender with SVMs



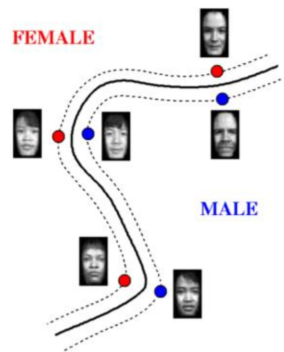
Processed faces



Support Vector Machines (SVM)

Example 6 Learning gender with SVMs

Training dataset: 1044 males, 713 females
Experiment with various kernels, select Gaussian RBF



Support Vector Machines (SVM)

Example 6 Learning gender with SVMs

Classifier	Error Rate		
	Overall	Male	Female
SVM with RBF kernel	**3.38%**	**2.05%**	**4.79%**
SVM with cubic polynomial kernel	**4.88%**	**4.21%**	**5.59%**
Large Ensemble of RBF	5.54%	4.59%	6.55%
Classical RBF	7.79%	6.89%	8.75%
Quadratic classifier	10.63%	9.44%	11.88%
Fisher linear discriminant	13.03%	12.31%	13.78%
Nearest neighbor	27.16%	26.53%	28.04%
Linear classifier	58.95%	58.47%	59.45%

Support Vector Machines (SVM)

Example 7 Two spirals benchmark problem

Carnegie Mellon AI Repository

$$\alpha = \frac{i\pi}{104\gamma} \qquad i = 1, 2, \cdots, n \qquad n \text{ : num of patterns}$$

γ : density

$$x = R\cos\alpha$$

R : radius

$$y = R\sin\alpha$$

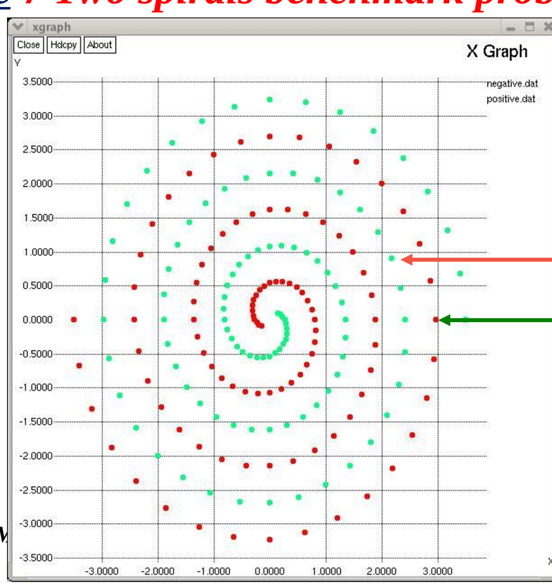
$$x^- = -x^+ \qquad y^- = -y^+$$

Data generation :

$$(x, y) \quad with \quad R = 3.5 \quad \gamma = 1.0 \quad n^- = 100 \quad n^+ = 100$$

Support Vector Machines (SVM)

Example 7 Two spirals benchmark problem



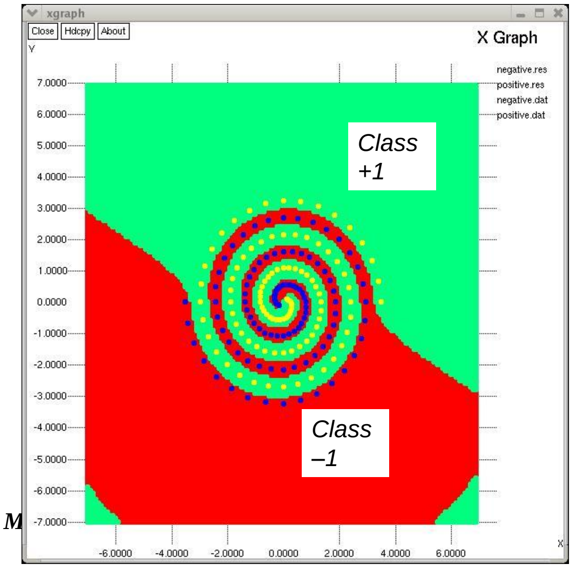
Class –1

Class +1

Support Vector Machines (SVM)

67

Example 7 Two spirals benchmark problem



IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom