# Riconoscimento e recupero dell'informazione per bioinformatica

## Classificatori discriminativi

Manuele Bicego

Corso di Laurea in Bioinformatica
Dipartimento di Informatica - Università di Verona

# Introduzione

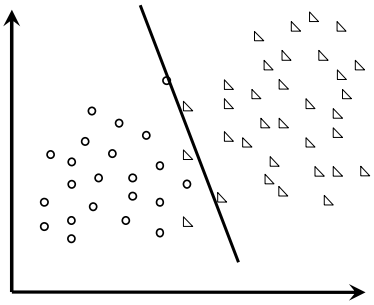
- ⇨ I classificatori generativi mirano a modellare le probabilità condizionali delle classi, da mettere assieme con quelle a priori per ottenere le posterior
  - ⇨ regola di decisione di Bayes
- ⇨ I classificatori discriminativi mirano ad ottenere direttamente il confine di decisione
  - ⇨ stimando direttamente le posterior
  - ⇨ o affidandosi a concetti "geometrici"
- ⇨ Approcci geometrici (possono essere trovati parallelismi tra questi e la regola di Bayes)
  - ⇨ Funzioni discriminanti lineari
  - ⇨ Support Vector Machines

2

# Funzioni discriminanti lineari

## Funzioni discriminanti lineari

⇨ Obiettivo è trovare la funzione che separa le due classi



⇨ Se la funzione discriminante è combinazione lineare delle varie features, allora si parla di discriminante lineare

4

# Funzioni discriminanti lineari

$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0$$

dove $\mathbf{x}=[x_1,...,x_n]$, $\mathbf{w}=[w_1,...,w_n]$ (pesi) e $w_0$ *bias* (soglia).

⇨ *Un campione* $\boldsymbol{x}_i$ *è classificato come appartenente a* $\omega_1$ *se* $\boldsymbol{w}^t\boldsymbol{x}_i+w_0>0$, *a* $\omega_2$ *se* $\boldsymbol{w}^t\boldsymbol{x}_i + w_0 < 0$,

⇨ Più genericamente, consideriamo come $\mathbf{w'} = [w_0,w_1,...,w_n]$ e $\mathbf{x'} =[1,x_1,...,x_n]$, quindi la regola di decisione diventa:
*Un campione* $\boldsymbol{x}_i$ *è classificato come appartenente a* $\omega_1$ *se* $\boldsymbol{w}'^t\boldsymbol{x}_i'>0$, *a* $\omega_2$ *se* $\boldsymbol{w}'^t\boldsymbol{x}_i' < 0$

5

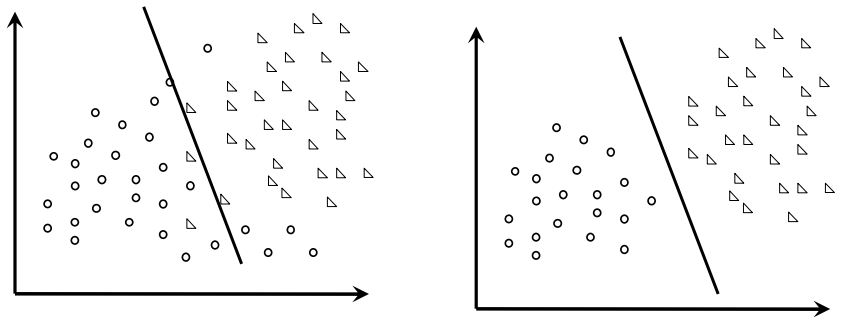
# Funzioni discriminanti lineari

⇨ Training: dato un training set (insieme di *m* campioni $x_1,...,x_m$, alcuni etichettati $\omega_1$ ed altri etichettati $\omega_2$), si vuole determinare il vettore dei pesi $\mathbf{w}$ e $w_0$ della funzione discriminante lineare

⇨ Un ragionevole approccio è la ricerca di un vettore di pesi tale che la probabilità di commettere errore sui campioni sia minima.

⇨ Se esiste un vettore di pesi tale da rendere nulla la probabilità di errore, allora i campioni si dicono *linearmente separabili*.

6



non linearmente separabili

linearmente separabili

7

# Due classi

- ⇨ L'obiettivo è quindi quello di calcolare tali pesi per cui
  $\mathbf{w}^t\mathbf{x}_i > 0$ per ogni $\mathbf{x}_i$ appartenente a $\omega_1$
  $\mathbf{w}^t\mathbf{x}_i < 0$ per ogni $\mathbf{x}_i$ appartenente a $\omega_2$
- ⇨ In quest'ultimo caso si può anche dire che $\mathbf{x}_i$ è classificato correttamente se $\mathbf{w}^t(-\mathbf{x}_i) > 0$.
- ⇨ Questo suggerisce una normalizzazione (cambiare il segno a tutti gli oggetti della classe 2) che semplifica il trattamento nel caso di due diverse classi, ossia il fatto che si possa solo trovare il vettore dei pesi tale che $\mathbf{w}^t\mathbf{x}_i>0$ per tutti i campioni a prescindere dalle classi.
- ⇨ Questo vettore è chiamato vettore separatore o vettore soluzione.

8

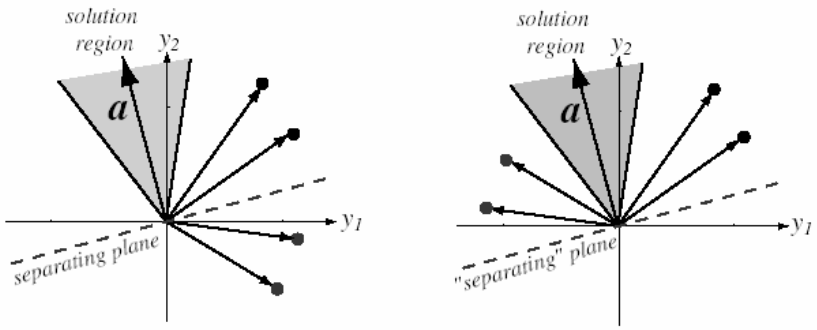



**FIGURE 5.8.** Four training samples (black for $\omega_1$, red for $\omega_2$) and the solution region in feature space. The figure on the left shows the raw data; the solution vectors leads to a plane that separates the patterns from the two categories. In the figure on the right, the red points have been "normalized"—that is, changed in sign. Now the solution vector leads to a plane that places all "normalized" points on the same side. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

9

- ⇨ Risulta chiaro, quindi, che se il vettore soluzione esiste non è unico.
- ⇨ Ci sono diversi modi per imporre requisiti addizionali per vincolare il vettore soluzione.
- ⇨ Uno potrebbe essere quello di cercare il vettore dei pesi di lunghezza unitaria che massimizzi la minima distanza dei campioni dal piano separatore.
- ⇨ Un'altra potrebbe essere quella di cercare il vettore dei pesi a lunghezza minima che soddisfi

$$\mathbf{w}^t \mathbf{x}_i \geq b, \quad \forall i$$

dove $b$ è una costante positiva chiamata *margine*.

10

# Determinazione dei pesi w

*Tecnica del Gradiente Discendente*

⇨ La tecnica del Gradiente Discendente è uno degli approcci più semplici per il calcolo di una funzione discriminante.

⇨ È un metodo iterativo di assestamento progressivo dei pesi che si basa sulla seguente proprietà:

*il vettore gradiente nello spazio W punta nella direzione di massimo scarto di una funzione da massimizzare/minimizzare*

11

# Determinazione dei pesi w

⇨ La procedura consiste nell'aggiornare il valore del vettore dei pesi al passo k+1 con un contributo proporzionale al modulo del gradiente stesso al passo precedente e può essere formulata come:

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \rho_k \nabla J(\mathbf{w})\Big|_{\mathbf{w}=\mathbf{w}(k)}$$

dove J(**w**) è una funzione di valutazione che deve essere minimizzata.

⇨ J(**w**) viene scelta in modo tale da raggiungere il minimo all'avvicinarsi di **w** alla soluzione ottima, ovvero convessa.

12

# Determinazione dei pesi w

⇨ Il minimo di J(**w**) si ottiene spostando **w** in direzione opposta al gradiente.

⇨ $\nabla$ è il simbolo dell'operatore gradiente, dato da:

$$\nabla J = \begin{bmatrix} \frac{\partial J}{\partial w_1} \\ \vdots \\ \vdots \\ \vdots \\ \frac{\partial J}{\partial w_n} \end{bmatrix}$$

⇨ $\rho_k$ è uno scalare opportuno che varia con l'iterazione, *k*, fissando l'"ampiezza" nella correzione.

13

# Determinazione dei pesi w

⇨ Chiaramente occorre scegliere un criterio di ottimalità J(**w**)

⇨ La scelta più ovvia è quella di definire un funzionale $\mathcal{J}(\mathbf{w};\mathbf{x}_1,\ldots,\mathbf{x}_n)$ rappresentato dal numero di campioni mal classificati da **w** (Metodo del Percettrone)

⇨ Siccome tale funzionale è costante a tratti, il metodo della discesa del gradiente non è molto adatto a tale problema.

⇨ Una scelta migliore per $J$ può essere perciò:

$$J(\mathbf{w}) = -\sum_{i \in X} \mathbf{w}^t \mathbf{x}_i$$

dove $X$ è l'insieme di punti classificati non correttamente da **w**.

⇨

14

# Determinazione dei pesi w

⇨ Geometricamente, $J(\mathbf{w})$ è proporzionale alla somma delle distanze dei campioni mal classificati dal confine di decisione.

⇨ Siccome la i-esima componente del gradiente di $J(\mathbf{w})$ è pari a $\partial J/\partial w_i$, si può osservare dall'eq. (5) che:

$$\nabla J = -\sum_{i \in X} \mathbf{x}_i$$

⇨ e quindi l'algoritmo di discesa del gradiente è

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \rho_k \cdot \sum_{i \in X} \mathbf{x}_i, \quad \text{con} \quad \rho_k = \frac{1}{k}$$

15

# Determinazione dei pesi w

⇨ Questo metodo è stato usato per simulare sia in modo *hardware* che *software* la prima rete neurale ed si è in seguito evoluto a più complesse versioni come, ad esempio il *perceptron* multilivello (vedi lezione sulle reti neurali)

⇨ Ci sono molti altri metodi per scegliere J(w) e per ottimizzarla:

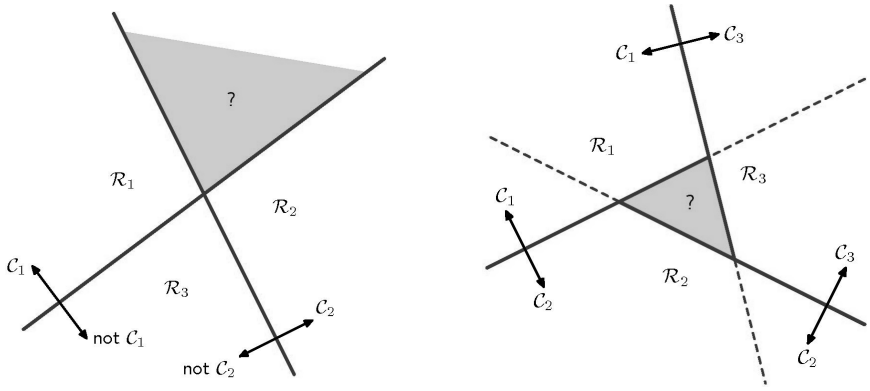
- ⇨ **Metodo del rilassamento**
- ⇨ **Metodo del MSE (minimun square error)**
- ⇨ **Metodo del Least MSE o Widrow-Hoff**
- ⇨ **Metodo di Ho-Kashyap**

16

# Caso multi classe

- ⇨ Nel caso di problemi a C classi, possono essere costruiti C-1 classificatori $g_i(\mathbf{x})$, uno per ogni classe $C_i$ contro *non-*$C_i$, chiamati classificatori *one-vs-rest*
- ⇨ Oppure si possono costruire C(C-1)/2 classificatori binari (*one-vs-one*), e quindi classificare un punto a maggioranza (fare un torneo)
- ⇨ Entrambi portano a zone di ambiguità.

17




18

# Funzioni discriminanti lineari generalizzate

⇨ Abbiamo visto che la funzione discriminante lineare può essere scritta come

$$g(x) = w_0 + \sum_{i=1}^{n} w_i x_i$$

dove $w_i$ è l'elemento del vettore dei pesi e $w_0$ è il peso di soglia.

⇨ Possiamo aggiungere altri termini in cui mettiamo i prodotti delle varie componenti di x

⇨ Esempio: classificatore discriminante quadratico

$$g(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n}\sum_{j=1}^{n} v_{ij} x_i x_j$$

19

⇨ Possiamo generalizzare il concetto scrivendo

$$g(x) = \sum_{i=1}^{d1} a_i \cdot y_i(x)$$

dove yi(x) sono funzioni arbitrarie su x

⇨ In questo caso, g(x) si chiama funzione discriminante lineare generalizzata

- ⇨ lineare rispetto a yi(x)
- ⇨ non lineare nello spazio originale di x

⇨ yi(x) rappresenta una sorta di "feature extraction"

- ⇨ solo che di solito si sale di dimensionalità, non si scende (come abbiamo visto nel caso della PCA)

20

Esempio:

⇨ consideriamo la funzione quadratica

$$g(x) = a_1 + a_2 x + a_3 x$$

⇨ Nella formula del discriminante lineare generalizzato, questo equivale a definire tre funzioni yi(x)

$$y_1(x) = 1$$
$$y_2(x) = x$$
$$y_3(x) = x^2$$

21

⇨ Quindi si passa da uno spazio monodimensionale ad uno spazio tridimensionale

⇨ Adesso la funzione g(x) è lineare nello spazio di y (cioè è un iperpiano nello spazio tridimensionale)

- ⇨ funzione semplice nello spazio di y
- ⇨ funzione complessa nello spazio originale di x

⇨ Idea alla base delle Support Vector Machines

22

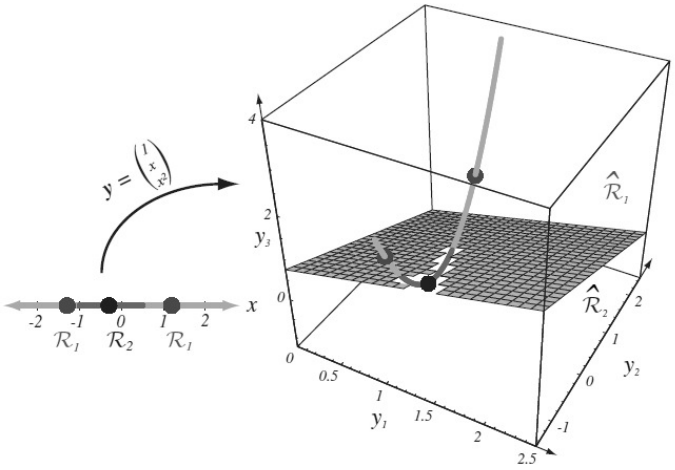



**FIGURE 5.5.** The mapping $\mathbf{y} = (1, x, x^2)^t$ takes a line and transforms it to a parabola in three dimensions. A plane splits the resulting $\mathbf{y}$-space into regions corresponding to two categories, and this in turn gives a nonsimply connected decision region in the one-dimensional $x$-space. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

# Support Vector Machines

(lucidi di Riccardo Gherardi)

# Outline

1. Overview
2. Support Vector Machines
   - The linear, separable case
   - Extension to non-separable data
   - Beyond linear classification with kernels
3. Getting practical
   - The “cookbook approach”
   - A concrete example
4. Conclusions

25

# Why SVMs?

1. Conceptually simple
2. Powerful and elegant
3. Fast

26

# Applications

Facial recognition
Content–based image retrieval
Facial expression classification
Hand–written text interpretation
3D object recognition
Texture Classification
Text classification
Traffic prediction
Disease identification
Gene sequencing
Protein folding
Weather forecasting
Earthquake prediction
Automated diagnosis
Many more…

Face recognition demo as seen in Viola, Jones, "Robust Real-time Object Detection", IJCV 2001

27

# Cronology

**1979**
Underlying theory developed, enunciation of the Structural Risk Minimization principle
Vapnik, "Estimation of Dependences Based on Empirical Data", Moscow, 1979.

**1992**
SVMs introduced in COLT-92
Boser, Guyon, Vapnik, "A training algorithm for optimal margin classifiers", Computational Learning Theory, Pittsburgh,1992.

*Vladimir Vapnik*

**1995**
Framework extended to non-separable problems
Cortes, Vapnik, "Support Vector Networks", *Machine Learning,* 1995.

**1999**
A fast algorithm for training and testing is proposed
Platt, "Fast training of support vector machines using sequential minimal optimization", in "Advances in Kernel Methods", MIT Press, Cambridge, MA, 1999.

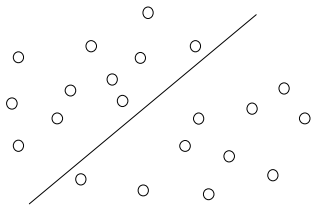
28

# Linear SVMs

- Binary classifier
- Data is linearly separable
  - exists a hyperplane that divides the classes
  - given a set $\{\mathbf{x}_i, y_i\}$ of tuples and their labels

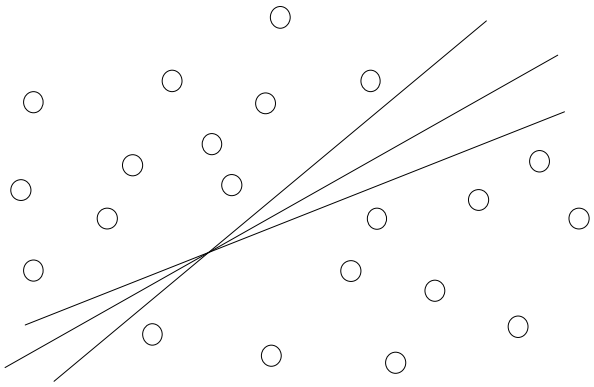
$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad \forall i \in \{1,...,m\}$$

$$\mathbf{w} \in R^n, \quad \mathbf{x}_i \in R^n, \quad b \in R, \quad y_i \in \{+1,-1\}$$



29

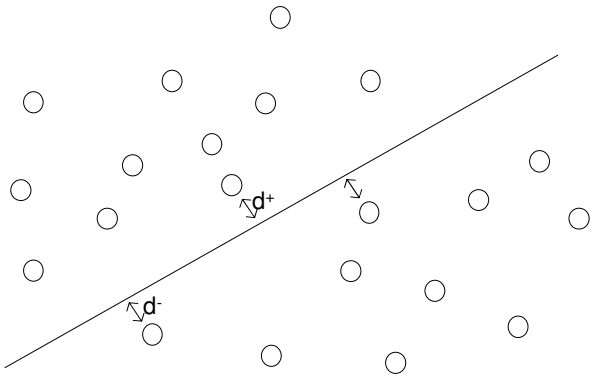
# Finding the hyperplane



What is the best separating hyperplane?

30

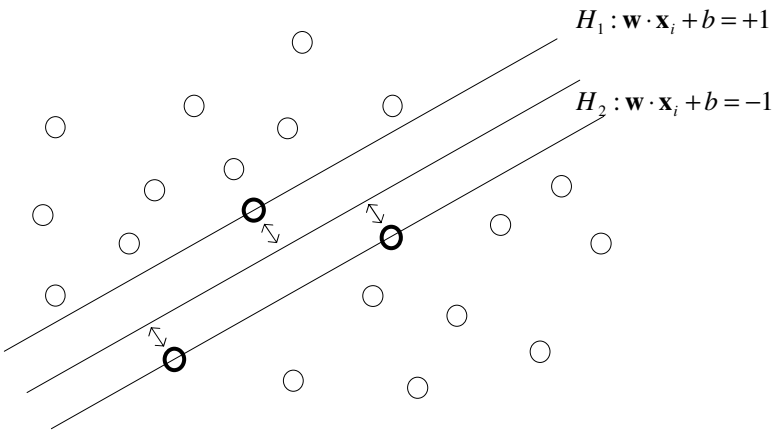
# Maximizing the margin




SVMs maximize the margin $d=d^+ + d^-$
$d^+$ ($d^-$) is the minimum distance to the nearest positive (negative) sample

31

# Support vectors




The samples which lie on the lines $H_1$ and $H_2$ are called *Support Vectors*

32

# Finding the hyperplane

If ||w|| is the euclidian norm of w, then:

$$d^+ = d^- = \frac{1}{\|\mathbf{w}\|}$$

Maximizing the margin $d = d^+ + d^- = \frac{2}{\|\mathbf{w}\|}$ is equivalent to minimize the quantity:

$$\min \frac{1}{2}\mathbf{w} \cdot \mathbf{w}$$

with $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad \forall i \in \{1,...,m\}$

33

# Solving for the hyperplane

The problem outlined is an instance of constrained quadratic optimization and hence can be solved using the technique of Lagrange multipliers

NOTE: the support vectors lie on the margin and all remaining examples of the training set are irrelevant (i.e., the contraints does not play a role in the optimization).

34

## Solving for the hyperplane

- **w** is a linear combination of all the support vectors
- An optimal solution always exists since in a quadratic optimization problem there are no local minima (convex problem)
- Problem complexity is proportional to the number of training vectors: naive implementations are computationally expensive
- Classification of new vector **x** is obtained computing

$$f(\mathbf{x}) = sign(\mathbf{w} \cdot \mathbf{x} + b)$$

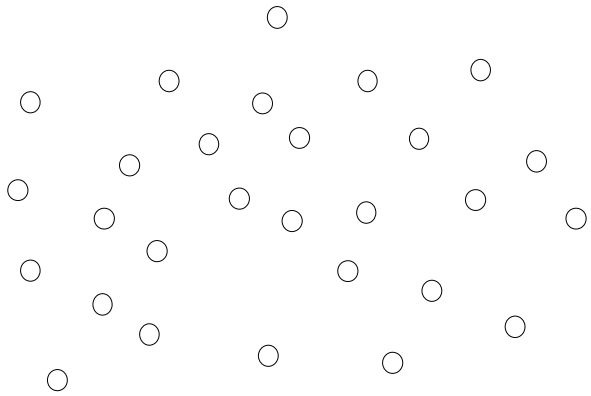
35

## Summary: linear SVMs

- Binary classifier
- Finds the best separating hyperplane
- The hyperplane is described by a small subset of training data, called support vectors
- The procedure is fast (polynomially bounded)
- We are guaranteed to find a optimal solution
- The method is statistical, not probabilistic

36

# Not linearly separable classes



What do we do if data
are not linearly separable?

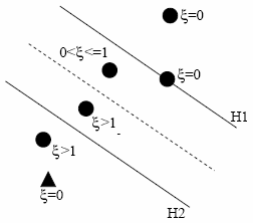
37

# Slack variables

- Quadratic optimization does not converge for non linearly separable data
- We modify the constraints adding “slack” variables, which enable vectors to cross the margin

$$y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i$$

- The value of the $\xi_i$ indicates the position of the vector with respect to the hyperplane




38

# Optimization with slack variables

The new objective function is

$$\min \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\cdot\left(\sum_{1}^{m}\xi_i\right)$$

$$y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i \geq 0$$

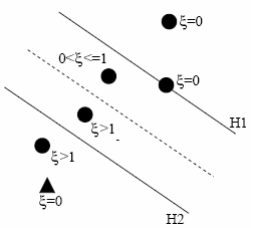
- *C* is a user-specified parameter which represents the cost for misclassified data
- *C* determines the sensitivity of the classifier to errors and its generalization performance

39

# Solution for NLS data

The constrained system can be solved in its dual form, solution is again

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

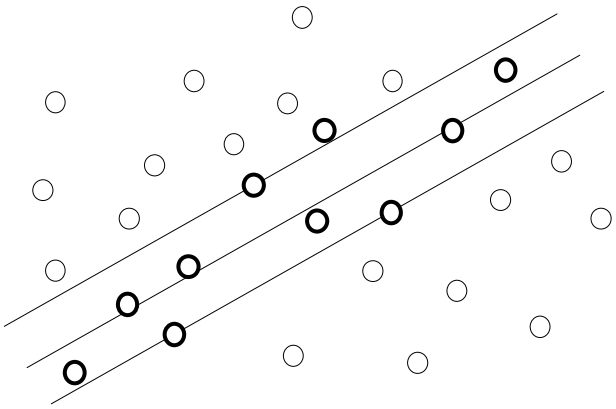



Values of ξ can be interpreted

| | |
|---|---|
| $\xi_i = 0$ | point is correctly classified |
| $0 < \xi_i < 1$ | correctly classified but outside $H_i$ |
| $\xi_i = 1$ | incorrectly classified, error |

40

# Support vectors in the NLS case

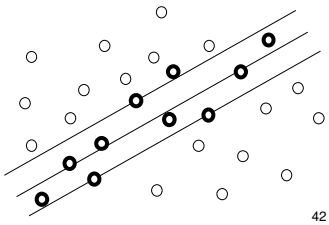


Support vectors are all points for which $\alpha \neq 0$
(misclassified data are also support vectors)

41

# Summary: SVMs in the NLS case

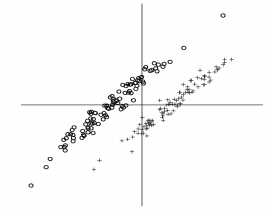
- Slack variables are introduced to allow vectors to cross the margin
- Support vectors contain every vector which lies on or *beyond* the margin
- We now need a free arbitrary parameter, the miclassification cost C

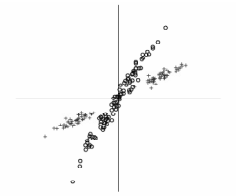


42

# Complex examples

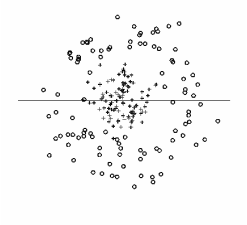
Parallel clouds

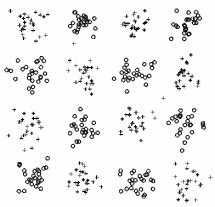




Split

Donut





Checkerboard

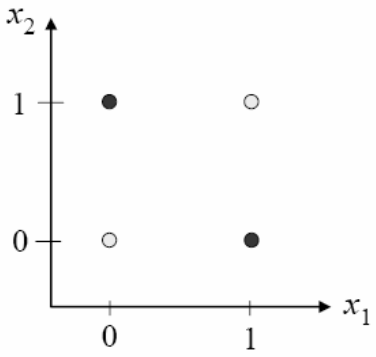
What do we do when linear hyperplanes don't suffice?

43

# A simpler example: XOR

Not linearly separable (Minsky & Papert '69)

*How can we tackle a problem like this with a SVM?*




44

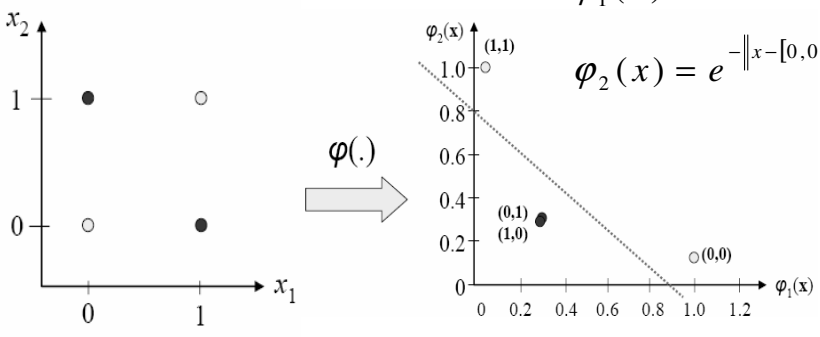
# Data mapping

Could it be separable when mapped into another space?

$$x = [x_1, x_2]$$

$$\varphi_1(x) = e^{-\left\|x-[1,1]^T\right\|}$$

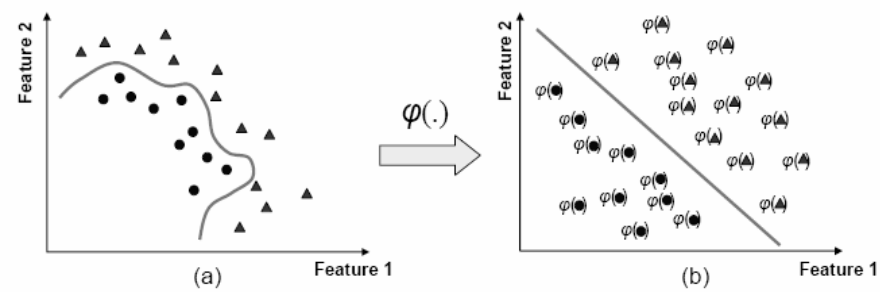
$$\varphi_2(x) = e^{-\left\|x-[0,0]^T\right\|}$$




45

# Data mapping: idea

A non linearly separable dataset can be mapped to another space (possibly of higher dimension), where a separating hyperplane exist




46

# Data mapping: another example

Let's map the space $\mathrm{X} = \{x, y, z\}$
into the higher dimensional space Z:

$$\varphi_1(\mathrm{X}) = x \quad \varphi_2(\mathrm{X}) = y \quad \varphi_3(\mathrm{X}) = z$$
$$\varphi_4(\mathrm{X}) = x^2 \quad \varphi_5(\mathrm{X}) = y^2 \quad \varphi_6(\mathrm{X}) = z^2$$
$$\varphi_7(\mathrm{X}) = xy \quad \varphi_8(\mathrm{X}) = xz \quad \varphi_9(\mathrm{X}) = yz$$

$$\mathrm{Z} = \left(\varphi_1(\mathrm{X}), \varphi_2(\mathrm{X}), \ldots, \varphi_9(\mathrm{X})\right)$$

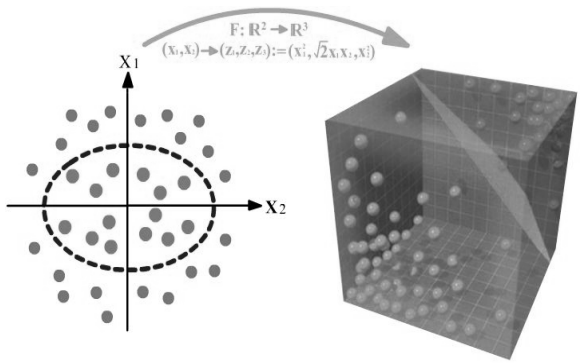
A linear classifier in the space Z corresponds to a polynomial one in the original space

47

# Mapping: one last example

The specified function $R^2 \rightarrow R^3$ maps the cartesian plane onto a 3D cone which intersection with the $x_1x_2$ plane gives the elliptical boundary

$$\varphi: \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$




48

# Problems of the mapping idea

1. Mappings can have huge dimensionality (even infinite)
2. Mappings are in general difficult to compute
3. It is not clear how to find the proper mapping that will separate the data

49

# Applying mapping to SVMs

- Crucial observation is that feature vectors in SVM training appear only in the form of dot products ($\mathbf{x}_i \cdot \mathbf{x}_j$)
- After applying a map φ, the training will be carried over the product $\varphi(x_i) \cdot \varphi(x_j)$
- If we could find a function *K* defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$$

  we could train the classifier without even ever bothering to explicitly compute the mapping φ.

- We call this function **Kernel** (and the technique *kernel trick*)

50

# The kernel concept

- Using a kernel, a SVM can work in a space of huge dimensionality while using the original algorithms
- The mapping to the target space is never calculated esplicitly, so it can be arbitrarily complicated
- We avoid the curse of dimensionality because the resulting classification algorithm is indipendent from the size of the target space
- Kernels can be seen as a problem specific module fitted in a general purpose algorithm

51

# Standard kernels

Linear $K(x,z) = \langle x, z \rangle$

Polynomial $K(x,z) = (\langle x, z \rangle + 1)^p$

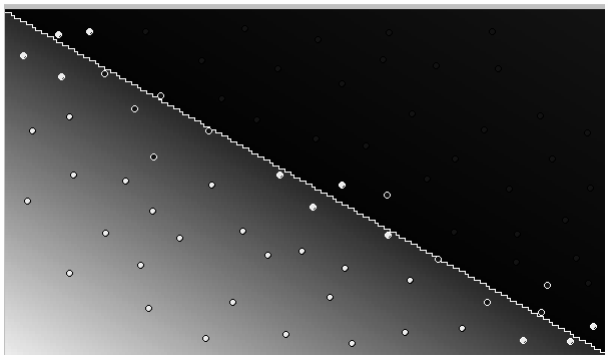
Radial basis functions $K(x,z) = e^{-\frac{\|x-z\|^2}{2\sigma^2}}$

Sigmoid $K(x,z) = \tanh(a\langle x, z \rangle + b)$

52

# Classification examples

Number of Support Vectors: **21** (-ve: 10, +ve: 11) Total number of points: 75



**Linear kernel**

Polynomial, deg 1
Polynomial, deg 2
Polynomial, deg 3
Polynomial, deg 4

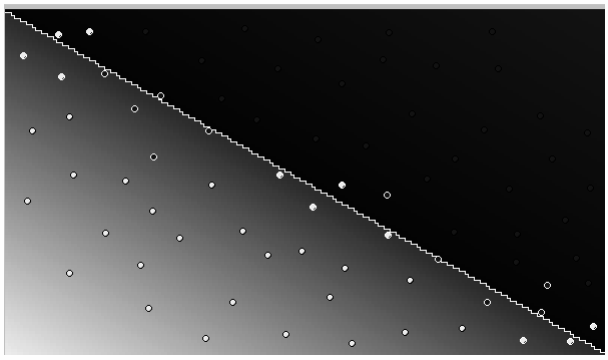
Polynomial deg 3, cost 100
Polynomial deg 3, cost 1000
Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5
Radial basis function, sigma 1
Radial basis function, sigma 2
Radial basis function, sigma 7

53

# Classification examples

Number of Support Vectors: **21** (-ve: 10, +ve: 11) Total number of points: 75



Linear kernel

**Polynomial, deg 1**
Polynomial, deg 2
Polynomial, deg 3
Polynomial, deg 4

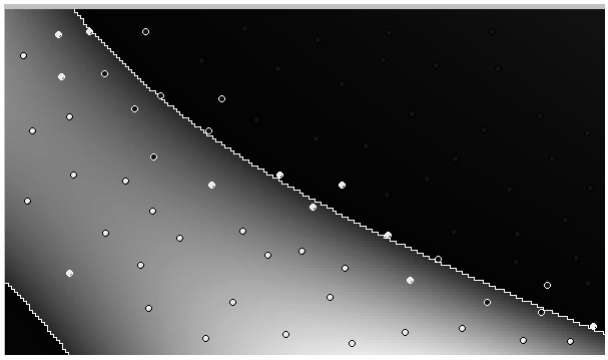
Polynomial deg 3, cost 100
Polynomial deg 3, cost 1000
Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5
Radial basis function, sigma 1
Radial basis function, sigma 2
Radial basis function, sigma 7

54

# Classification examples

Number of Support Vectors: **22** (-ve: 11, +ve: 11) Total number of points: 75



Linear kernel

Polynomial, deg 1
**Polynomial, deg 2**
Polynomial, deg 3
Polynomial, deg 4

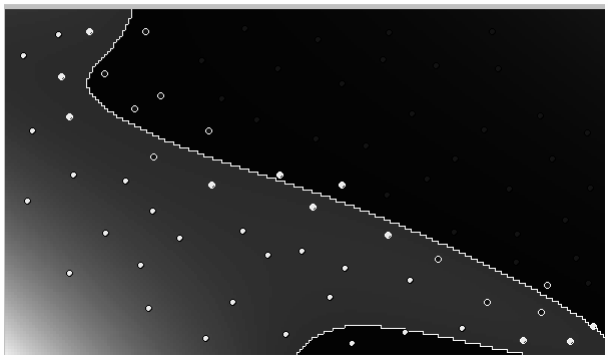
Polynomial deg 3, cost 100
Polynomial deg 3, cost 1000
Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5
Radial basis function, sigma 1
Radial basis function, sigma 2
Radial basis function, sigma 7

55

# Classification examples

Number of Support Vectors: **21** (-ve: 10, +ve: 11) Total number of points: 75



Linear kernel

Polynomial, deg 1
Polynomial, deg 2
**Polynomial, deg 3**
Polynomial, deg 4

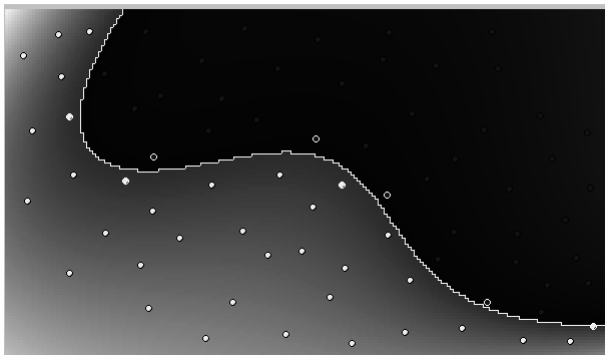
Polynomial deg 3, cost 100
Polynomial deg 3, cost 1000
Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5
Radial basis function, sigma 1
Radial basis function, sigma 2
Radial basis function, sigma 7

56

# Classification examples

Number of Support Vectors: **8** (-ve: 4, +ve: 4) Total number of points: 75



Linear kernel

Polynomial, deg 1
Polynomial, deg 2
Polynomial, deg 3
**Polynomial, deg 4**

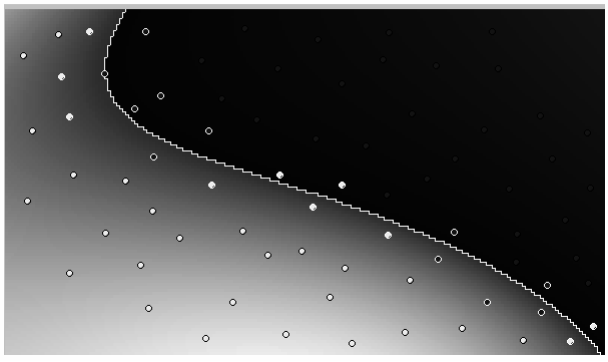
Polynomial deg 3, cost 100
Polynomial deg 3, cost 1000
Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5
Radial basis function, sigma 1
Radial basis function, sigma 2
Radial basis function, sigma 7

57

# Classification examples

Number of Support Vectors: **21** (-ve: 11, +ve: 10) Total number of points: 75



Linear kernel

Polynomial, deg 1
Polynomial, deg 2
Polynomial, deg 3
Polynomial, deg 4

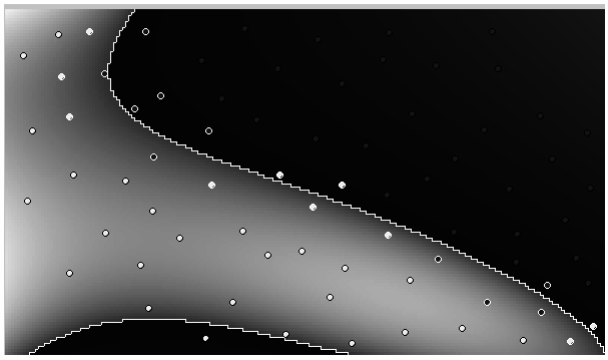
**Polynomial deg 3, cost 100**
Polynomial deg 3, cost 1000
Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5
Radial basis function, sigma 1
Radial basis function, sigma 2
Radial basis function, sigma 7

58

# Classification examples

Number of Support Vectors: **20** (-ve: 10, +ve: 10) Total number of points: 75



Linear kernel

Polynomial, deg 1
Polynomial, deg 2
Polynomial, deg 3
Polynomial, deg 4

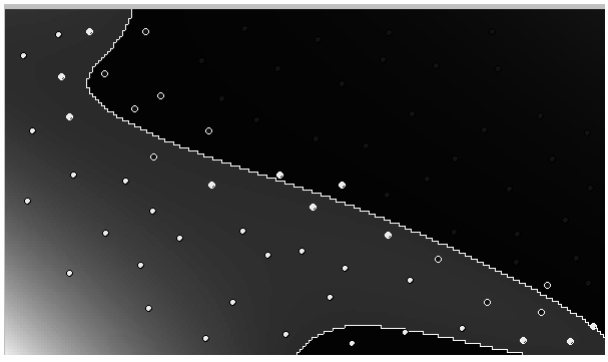
Polynomial deg 3, cost 100
**Polynomial deg 3, cost 1000**
Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5
Radial basis function, sigma 1
Radial basis function, sigma 2
Radial basis function, sigma 7

59

# Classification examples

Number of Support Vectors: **21** (-ve: 10, +ve: 11) Total number of points: 75



Linear kernel

Polynomial, deg 1
Polynomial, deg 2
Polynomial, deg 3
Polynomial, deg 4

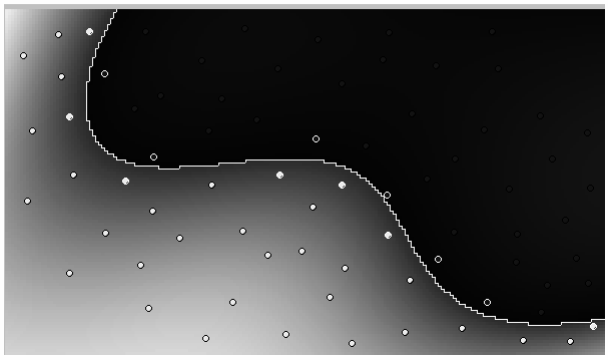
Polynomial deg 3, cost 100
Polynomial deg 3, cost 1000
**Polynomial deg 3, cost 10000**

Radial basis function, sigma 0.5
Radial basis function, sigma 1
Radial basis function, sigma 2
Radial basis function, sigma 7

60

# Classification examples

Number of Support Vectors: **13** (-ve: 6, +ve: 7) Total number of points: 75



Linear kernel

Polynomial, deg 1
Polynomial, deg 2
Polynomial, deg 3
Polynomial, deg 4

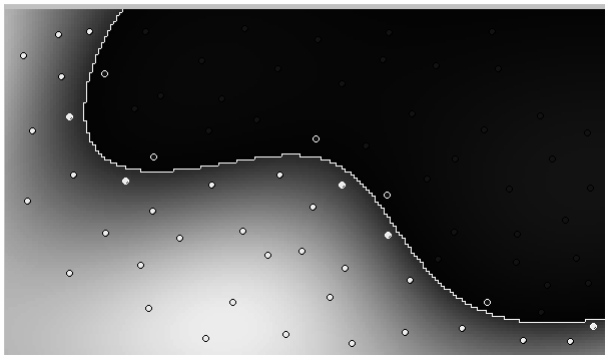
Polynomial deg 3, cost 100
Polynomial deg 3, cost 1000
Polynomial deg 3, cost 10000

**Radial basis function, sigma 0.5**
Radial basis function, sigma 1
Radial basis function, sigma 2
Radial basis function, sigma 7

61

# Classification examples

Number of Support Vectors: **10** (-ve: 5, +ve: 5) Total number of points: 75



Linear kernel

Polynomial, deg 1
Polynomial, deg 2
Polynomial, deg 3
Polynomial, deg 4

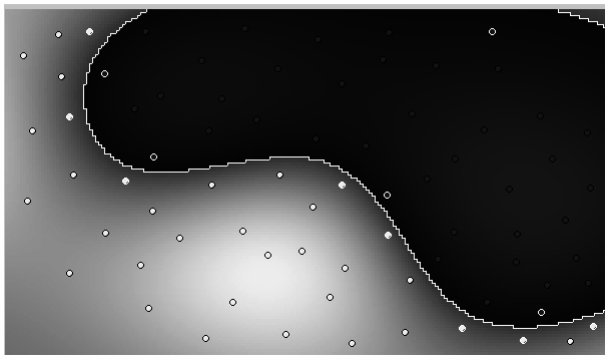
Polynomial deg 3, cost 100
Polynomial deg 3, cost 1000
Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5
**Radial basis function, sigma 1**
Radial basis function, sigma 2
Radial basis function, sigma 7

62

# Classification examples

Number of Support Vectors: **13** (-ve: 5, +ve: 8) Total number of points: 75



Linear kernel

Polynomial, deg 1
Polynomial, deg 2
Polynomial, deg 3
Polynomial, deg 4

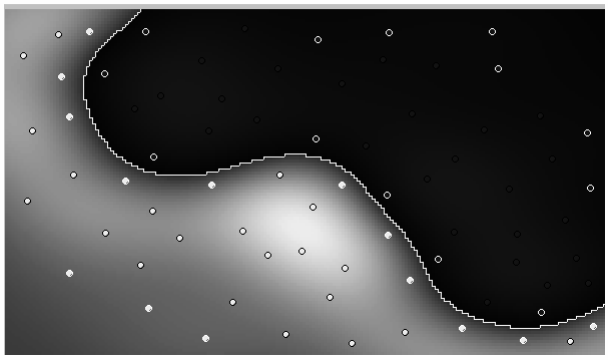
Polynomial deg 3, cost 100
Polynomial deg 3, cost 1000
Polynomial deg 3, cost 10000

Radial basis function, sigma 0.5
Radial basis function, sigma 1
**Radial basis function, sigma 2**
Radial basis function, sigma 7

63

# Classification examples

Number of Support Vectors: **27** (-ve: 13, +ve: 14) Total number of points: 75



Linear kernel

Polynomial, deg 1
Polynomial, deg 2
Polynomial, deg 3
Polynomial, deg 4

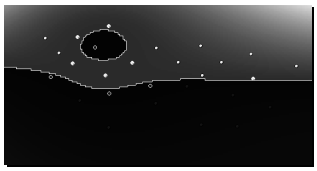
Polynomial deg 3, cost 100
Polynomial deg 3, cost 1000
Polynomial deg 3, cost 10000

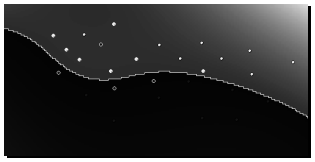
Radial basis function, sigma 0.5
Radial basis function, sigma 1
Radial basis function, sigma 2
**Radial basis function, sigma 7**

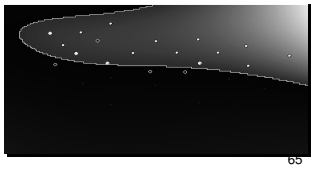
64

# Comments

- Kernel selection and parameter tuning are critical
- Cost C has a huge impact on the generalization ability
- Lowering degree or sigma can avoid overfitting
- Number of support vectors is a measure of generalization performance



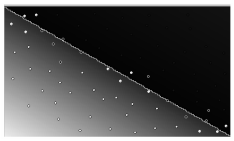




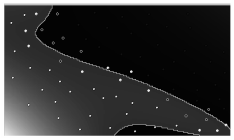
65

# Kernel selection

Linear kernel



- Used when the feature space is huge (for example in text classification, which uses individual word counts as features)
- Shown to be a special case of the RBF kernel
- No additional parameters



Polynomial

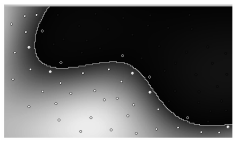
- Has numerical difficulties approaching 0 or infinity
- A good choice for well known and well conditioned tasks
- One additional parameter (degree p)

66

# Kernel selection

Radial basis functions

- Indicated in general as the best choice in the literature
- One additional parameter (sigma $\sigma$)



Sigmoid

- Two additional parameters (a and b)
- For some values of a and b, the kernel doesn't satisfy the Mercer condition
- From neural networks
- Not recommended in the literature

*Choosing the right kernel is still an art!*

67

# Cookbook approach

1. Conduct simple scaling on the data
2. Consider RBF kernel
3. Use cross-validation to find the best parameter *C* and $\sigma$
4. Use the best *C* and $\sigma$ to train the whole training set
5. Test

68

# Cookbook problems

- Parameter search can be very time consuming

→Solution: conduct parameter search hierarchically

- RBF kernels are sometimes subject to overfitting

→Solution: use high degree polynomials kernels

- Parameter search must be repeated for every chosen features; there no reuse of computations

→Solution: compare features on random subsets of the entire dataset to contain computational cost

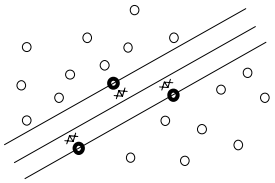
- Search ranges for C and $\sigma$ are tricky to choose

→Solution: literature suggests using exponentially growing values like $C = 2^{[-5..15]}$ and $\sigma = 2^{[-15..5]}$

69

# Summary

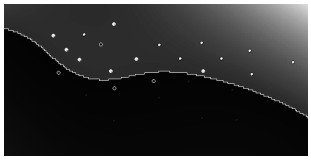
- Support Vector Machines, two key ideas
  - Margin maximization
  - Kernel trick



- Training
  - a quadratic optimization problem
  - always convergent to the optimal solution
  - solvable in polynomial time

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

- Free parameters
  - The cost C
  - The kernel type
  - The kernel parameters



70

# Advantages

- The SVM theory is an elegant and highly principled
- SVMs have a simple geometric interpretation
- The use of kernels provides a efficient solution to non-linear classification and dispels the ”curse of dimensionality”
- Convergence to the solution is guaranteed
- Support vectors give a compact representation of the entire dataset; their number is a measure of the generalization performance

71

# Disadvantages

- Kernel and parameter choice is crucial
- Training can sometimes be tricky and time consuming
- Training is not incremental; the whole dataset must be processed for every new addition
- There are no optimized extension to multi-class problems: a problem with N classes requires N classifiers

72

# References

**An introduction to Support Vector Machines**
(and other kernel-based learning methods)
N. Cristianini and J. Shawe-Taylor
Cambridge University Press (2000 )

**Support Vector and Kernel Machines**
Tutorial presented at ICML, Nello Cristianini

**A tutorial on support vector machines for pattern recognition**
C. Burges, Data Mining and Knowledge Discovery, Vol 2(2), June 1998

http://www.kernel-machines.org/
http://www.support-vector.net/
http://www.google.com/

73