L

COMPUTATIONAL MODELS FOR DATA ANALYSIS

Support Vector Machines

Alessandra Giordani

Department of information and communication technology
University of Trento
Email: agiordani@dit.unitn.it

UNIVERSITAS ATHESINA STUDIORUM

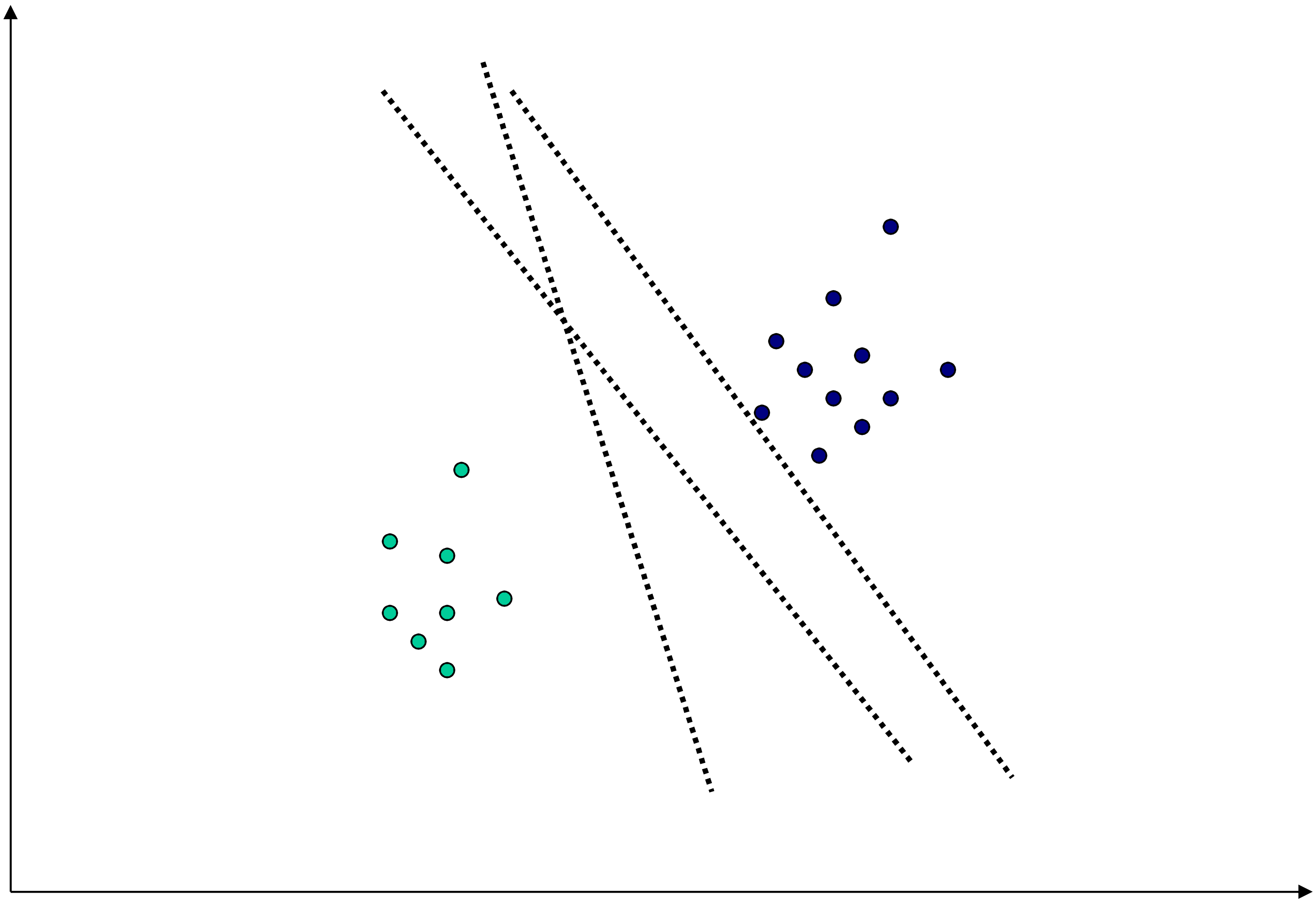
Course Schedule

- April 16: 15:45 - 18:15
- May 7: 15:45 - 18:15
- May 14: 15:45 - 18:15
- May 16: 14:30 - 17:00
- May 21: 15:45 - 18:15
- May 23: 14:30 - 17:00
- May 28: 14:30 - 17:00
- ~~May 30 21: 14:30 - 17:00~~

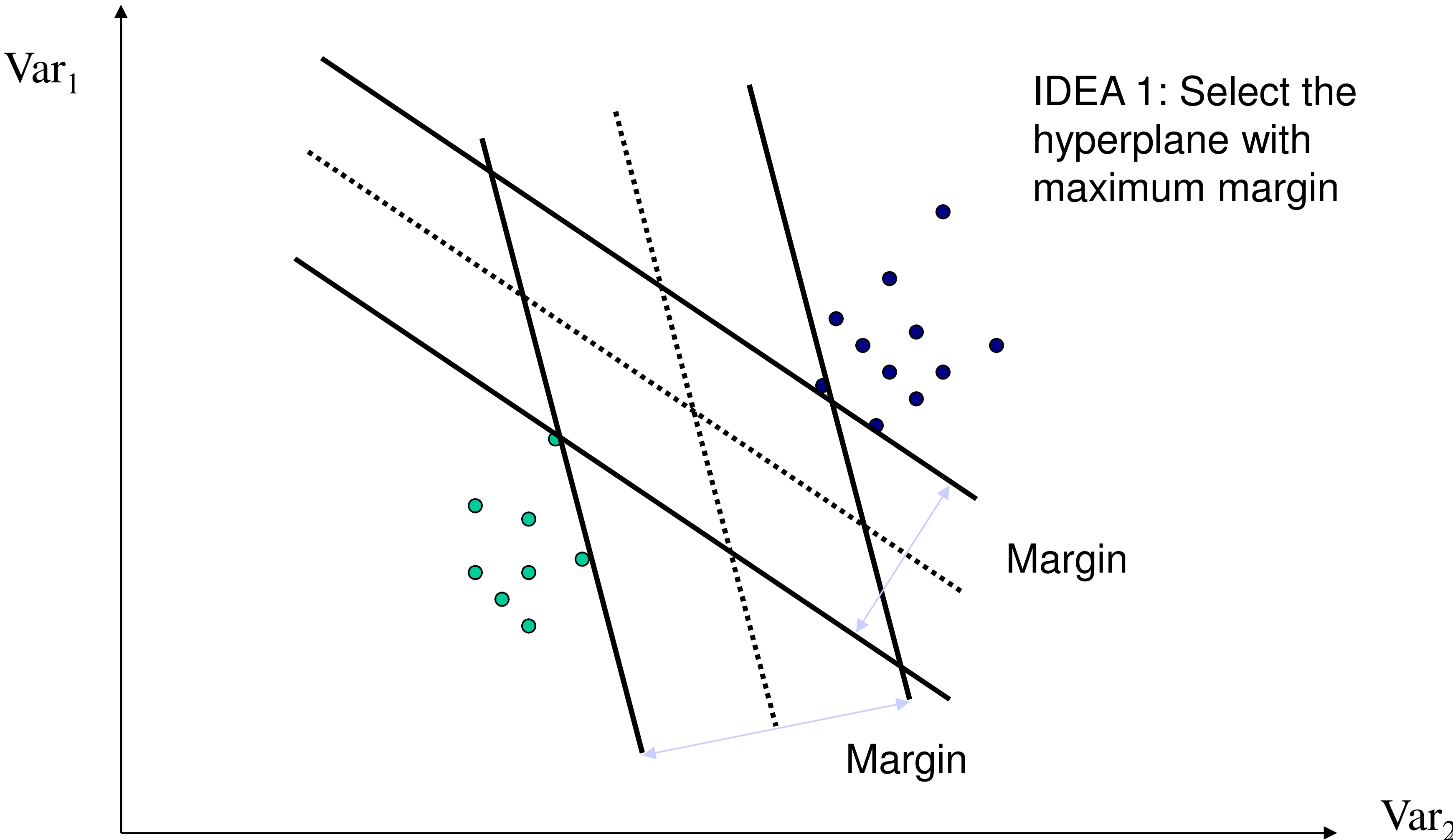
Summary

- Support Vector Machines
  - *Hard-margin SVMs*
  - *Soft-margin SVMs*

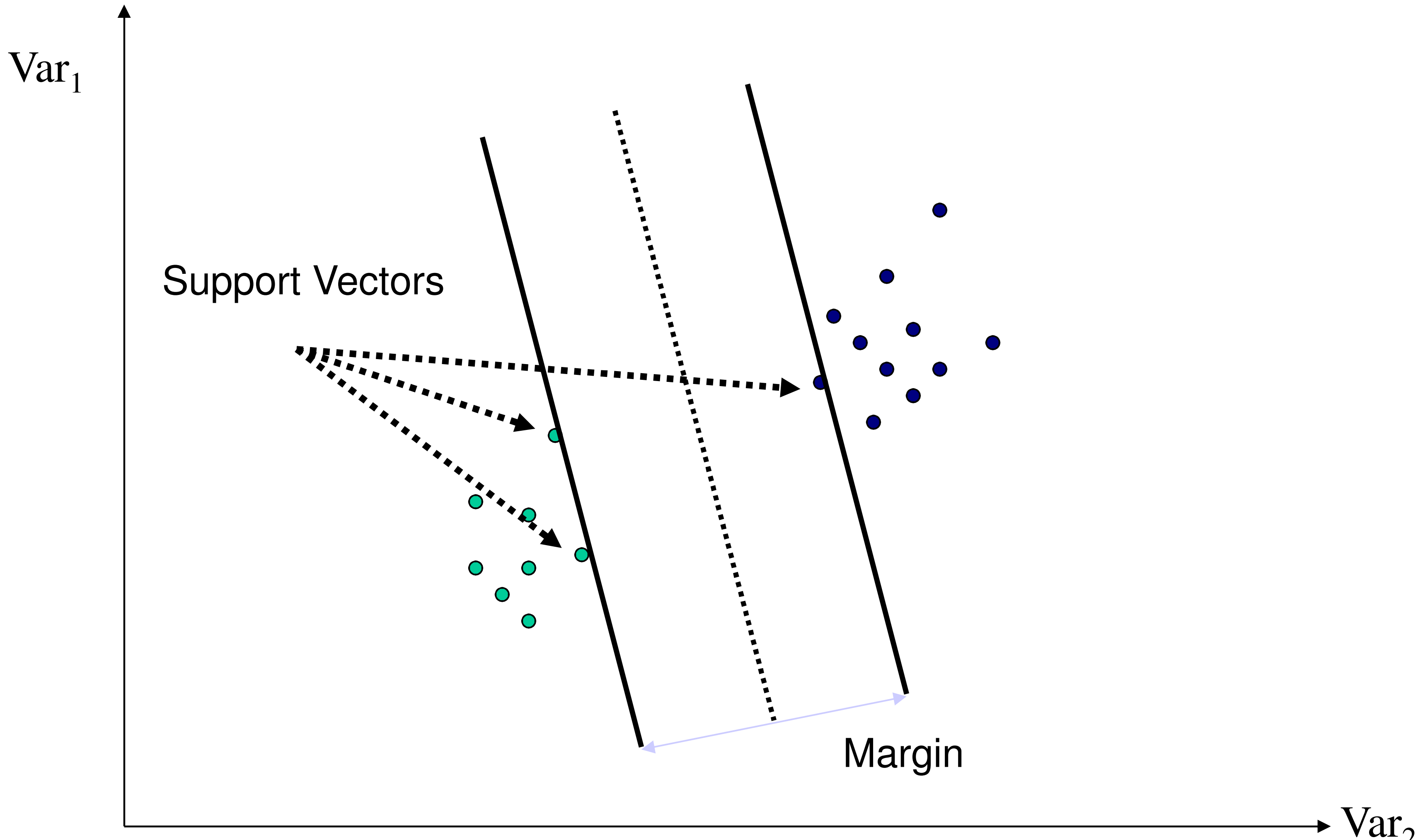
Which hyperplane choose?



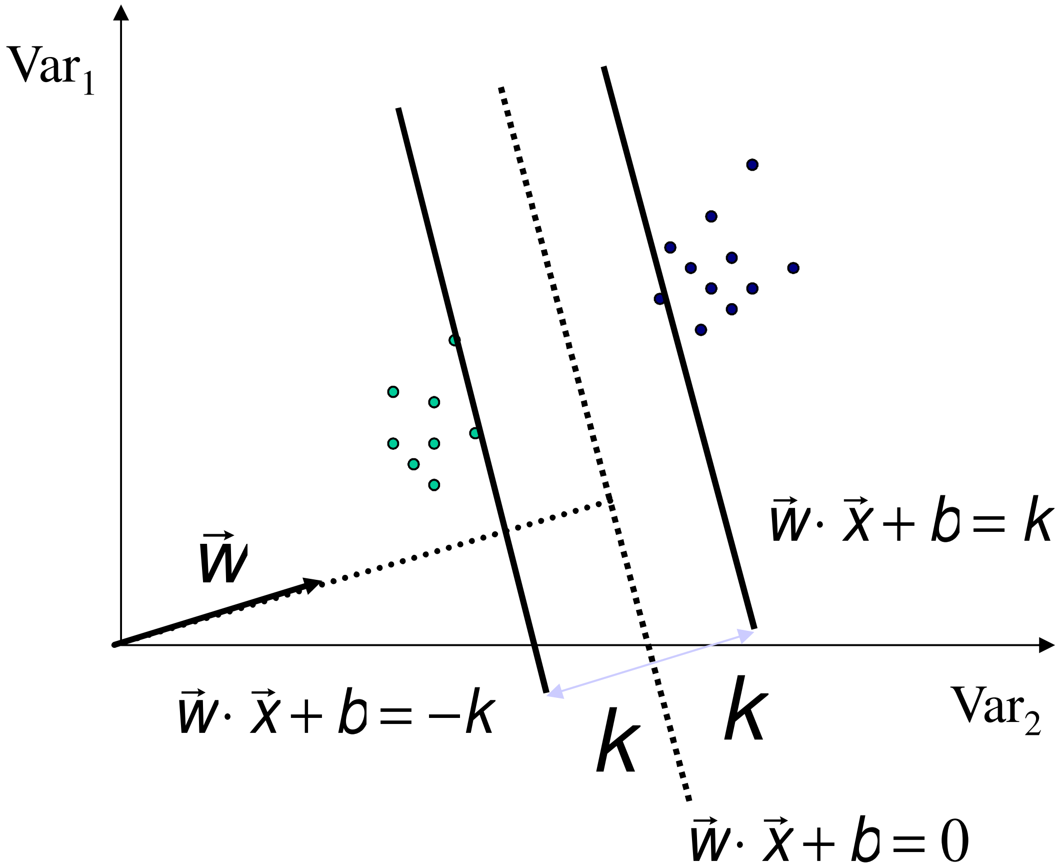
Classifier with a Maximum Margin



Support Vector

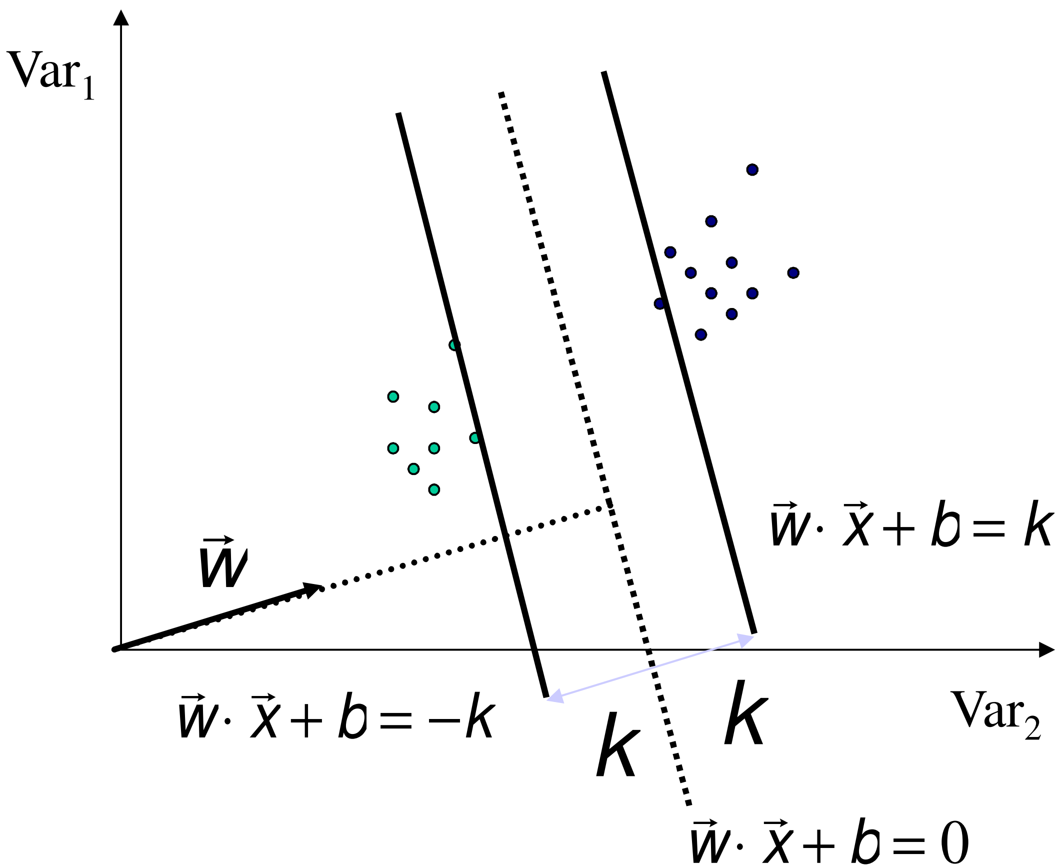


Support Vector Machine Classifiers



The margin is equal to $\dfrac{2|k|}{\|w\|}$

Support Vector Machines



The margin is equal to $\frac{2|k|}{\|w\|}$

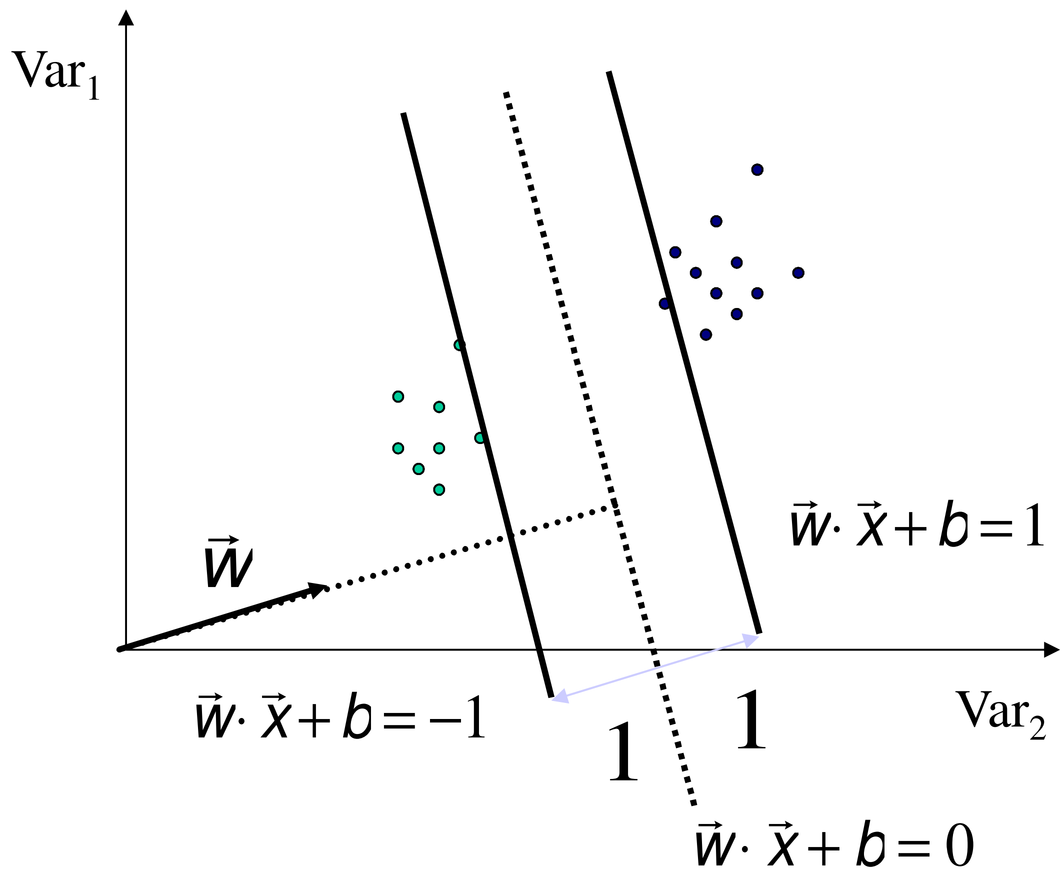
We need to solve

$$\max \frac{2|k|}{\|\vec{w}\|}$$

$$\vec{w}\cdot\vec{x}+b\geq +k,\ \text{ if } \vec{x} \text{ is positive}$$

$$\vec{w}\cdot\vec{x}+b\leq -k,\ \text{ if } \vec{x} \text{ is negative}$$

Support Vector Machines



There is a scale for which *k=1*.

The problem transforms in:

$$\max \frac{2}{\|\vec{w}\|}$$

$$\vec{w}\cdot\vec{x}+b \geq +1, \quad \text{if } \vec{x} \text{ is positive}$$

$$\vec{w}\cdot\vec{x}+b \leq -1, \quad \text{if } \vec{x} \text{ is negative}$$

Final Formulation

$$\max \frac{2}{\|\vec{w}\|}$$
$$\vec{w}\cdot\vec{x}_i + b \geq +1, \quad y_i = 1$$
$$\vec{w}\cdot\vec{x}_i + b \leq -1, \quad y_i = -1$$

$$\Rightarrow \quad \max \frac{2}{\|\vec{w}\|} \quad \Rightarrow$$
$$y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1$$

$$\Rightarrow \quad \min \frac{\|\vec{w}\|}{2} \quad \Rightarrow \quad \min \frac{\|\vec{w}\|^2}{2}$$
$$y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1 \qquad y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1$$

Optimization Problem

- Optimal Hyperplane:
  - Minimize $\tau(\vec{w}) = \frac{1}{2}\|\vec{w}\|^2$
  - Subject to $y_i((\vec{w} \cdot \vec{x}_i) + b) \geq 1, i = 1, \ldots, m$
- The dual problem is simpler

Warning!

- On the graphical examples, we always consider normalized hyperplane (hyperplanes with normalized gradient)
- *b* in this case is exactly the distance of the hyperplane from the origin
- So if we have an equation not normalized we may have

  $\vec{x} \cdot \vec{w} + b = 0$ with $\vec{x} = (x, y)$ and $\vec{w} = (1,1)$
- and *b* is not the distance

Warning!

- Let us consider a normalized gradient

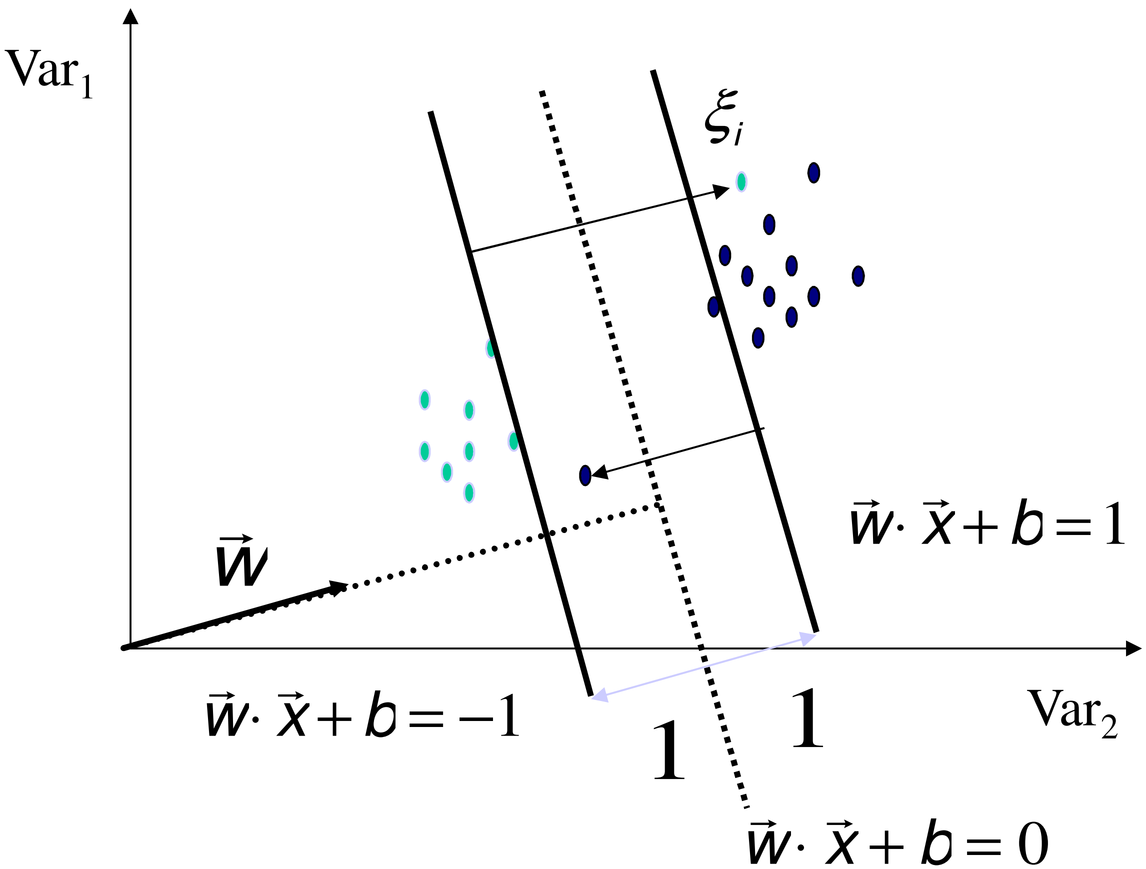
$$\vec{w} = \left(1/\sqrt{2}, 1/\sqrt{2}\right)$$

$$(x, y) \cdot \left(1/\sqrt{2}, 1/\sqrt{2}\right) + b = 0 \Rightarrow x/\sqrt{2} + y/\sqrt{2} = -b$$

$$\Rightarrow y = -x - b\sqrt{2}$$

- Now we see that *-b* is exactly the distance.
- For *x =0,* we have the intersection with $-b\sqrt{2}$. This distance projected on $\vec{w}$ is *-b*

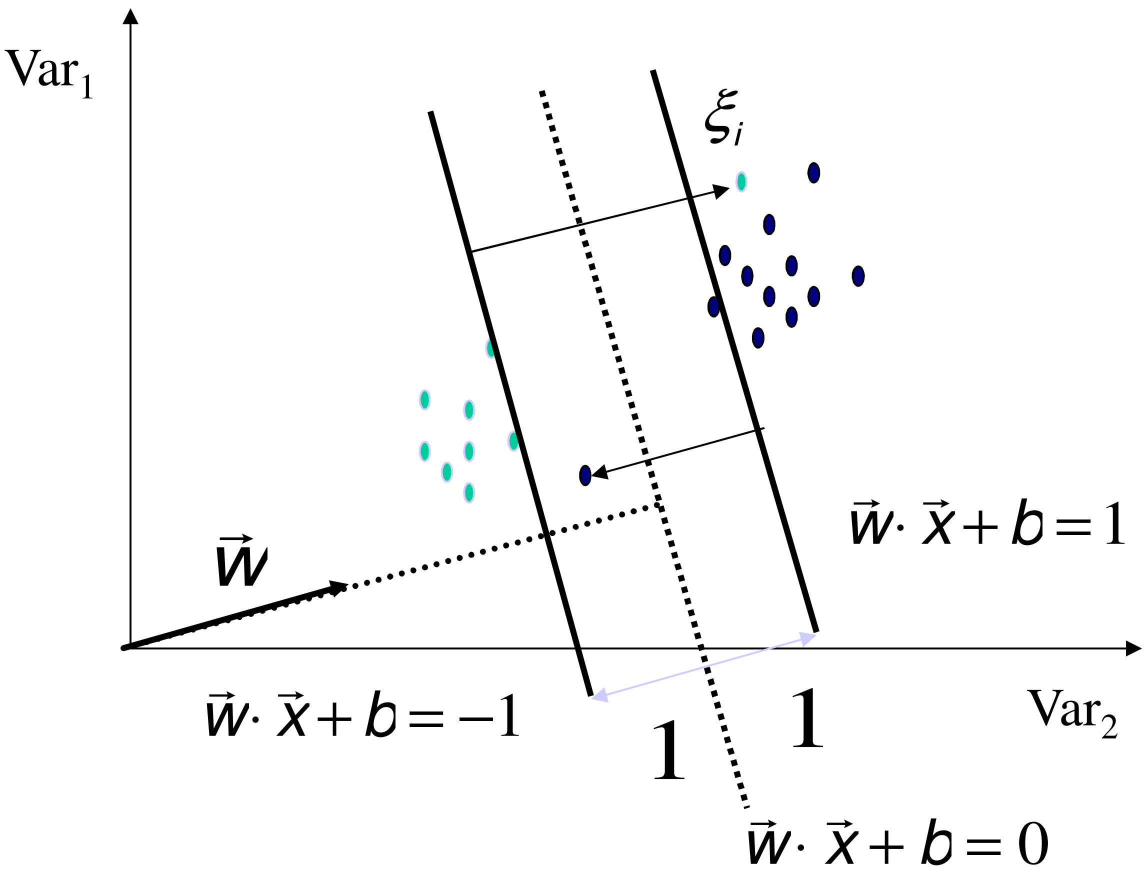
Soft Margin SVMs



ξ_i slack variables are added

Some errors are allowed but they should penalize the objective function

Soft Margin SVMs



The new constraints are

$$y_i(\vec{w}\cdot\vec{x}_i+b)\geq 1-\xi_i$$
$$\forall\vec{x}_i \text{ where } \xi_i\geq 0$$

The objective function penalizes the incorrect classified examples

$$\min\frac{1}{2}\|\vec{w}\|^2+C\sum_i\xi_i$$

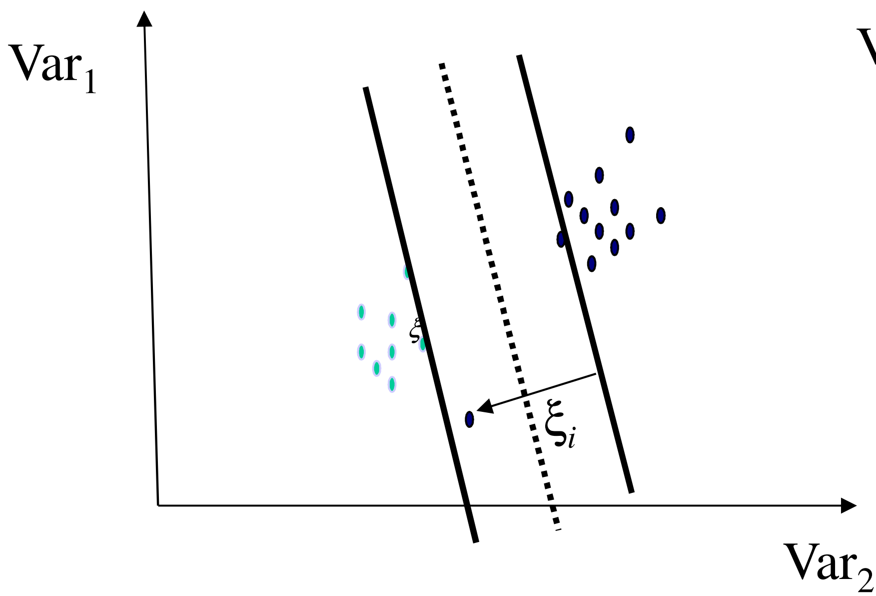
C is the trade-off between margin and the error

Soft Margin Support Vector Machines

$$\min \frac{1}{2} \|\vec{w}\|^2 + C\sum_i \xi_i \qquad \begin{array}{l} y_i(\vec{w}\cdot \vec{x}_i + b) \geq 1-\xi_i \quad \forall \vec{x}_i \\ \xi_i \geq 0 \end{array}$$

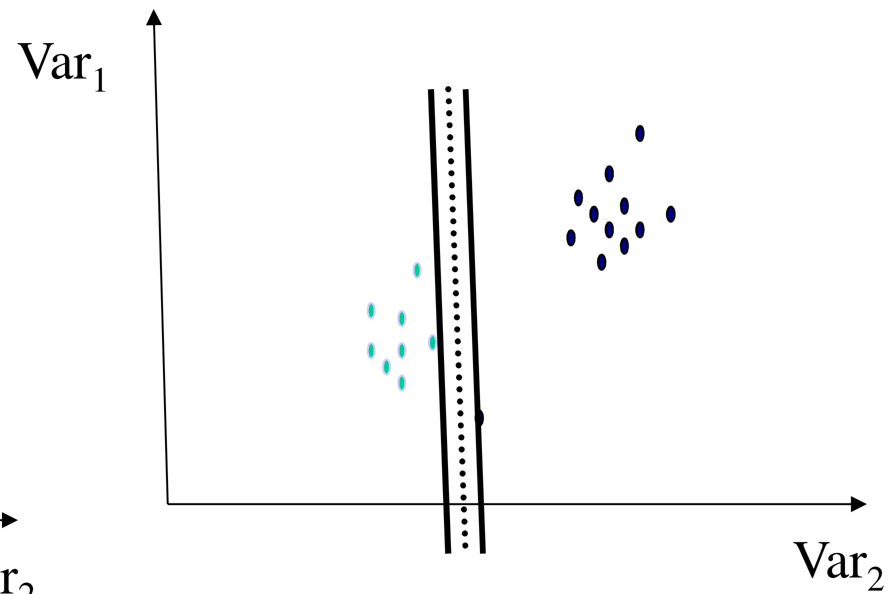
- The algorithm tries to keep ξ_i low and maximize the margin
- NB: The number of error is not directly minimized (NP-complete problem); the distances from the hyperplane are minimized
- If $C \to \infty$, the solution tends to the one of the *hard-margin* algorithm
- *Attention !!!:* if $C = 0$ we get $\|\vec{w}\| = 0$, since $y_i b \geq 1-\xi_i \quad \forall \vec{x}_i$
- If C increases the number of error decreases. When C tends to infinite the number of errors must be 0, i.e. the *hard-margin* formulation

Robusteness of *Soft* vs. *Hard Margin* SVMs



$\vec{w}\cdot\vec{x}+b=0$

Soft Margin SVM



$\vec{w}\cdot\vec{x}+b=0$

Hard Margin SVM

Soft vs Hard Margin SVMs

- *Soft-Margin* has ever a solution
- Soft-Margin is more robust to odd examples
- *Hard-Margin* does not require parameters

Parameters

$$\min \frac{1}{2} \| \vec{w} \|^2 + C \sum_i \xi_i = \min \frac{1}{2} \| \vec{w} \|^2 + C^+ \sum_i \xi_i^+ + C^- \sum_i \xi_i^-$$

$$= \min \frac{1}{2} \| \vec{w} \|^2 + C \left(J \sum_i \xi_i^+ + \sum_i \xi_i^- \right)$$

- C: trade-off parameter
- J: cost factor

Theoretical Justification

Definition of Training Set error

- Training Data

$$f : R^N \rightarrow \{\pm 1\} \qquad (\vec{x}_1, y_1), \ldots, (\vec{x}_m, y_m) \in R^N \times \{\pm 1\}$$

- Empirical Risk (error)

$$R_{emp}[f] = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \left| f(\vec{x}_i) - y_i \right|$$

- Risk (error)

$$R[f] = \int \frac{1}{2} \left| f(\vec{x}) - y \right| dP(\vec{x}, y)$$

Error Characterization (part 1)

- From PAC-learning Theory (***Vapnik***):

$$R(\alpha) \leq R_{emp}(\alpha) + \varphi\left(\frac{d}{m}, \frac{\log(\delta)}{m}\right)$$

$$\varphi\left(\frac{d}{m}, \frac{\log(\delta)}{m}\right) = \sqrt{\frac{d(\log\frac{2m}{d}+1)-\log(\frac{\delta}{4})}{m}}$$

where d is theVC-dimension, m is the number of examples, δ is a bound on the probability to get such error and α is a classifier parameter.

There are many versions for different bounds

Theorem 2.11 *(Vapnik and Chervonenkis, [Vapnik, 1995])*
Let H be a hypothesis space having VC dimension d. For any probability distribution D on $X \times \{-1, 1\}$, with probability $1-\delta$ over m random examples S, any hypothesis $h \in H$ that is consistent with S has error no more than

$$error(h) \leq \epsilon(m, H, \delta) = \frac{2}{m}\left(d \times ln\frac{2e \times m}{d} + ln\frac{2}{\delta}\right),$$

provided that $d \leq m$ and $m \geq 2/\epsilon$.

Error Characterization (part 2)

Lemma 1. [Vapnik, 1982] *Consider hyperplanes* $h(\vec{d}) = sign\{\vec{w} \cdot \vec{d} + b\}$ *as hypotheses. If all example vectors* $\vec{d}_i$ *are contained in a ball of radius* R *and it is required that for all examples* $\vec{d}_i$

$$|\vec{w} \cdot \vec{d}_i + b| \geq 1, \; with \; ||\vec{w}|| = A \tag{5}$$

then this set of hyperplane has a VCdim d *bounded by*

$$d \leq min([R^2 A^2], n) + 1 \tag{6}$$

Ranking, Regression and Multiclassification

The Ranking SVM
[Herbrich et al. 1999, 2000; Joachims et al. 2002]

- The aim is to classify instance pairs as correctly ranked or incorrectly ranked
  - This turns an ordinal regression problem back into a binary classification problem
- We want a ranking function f such that

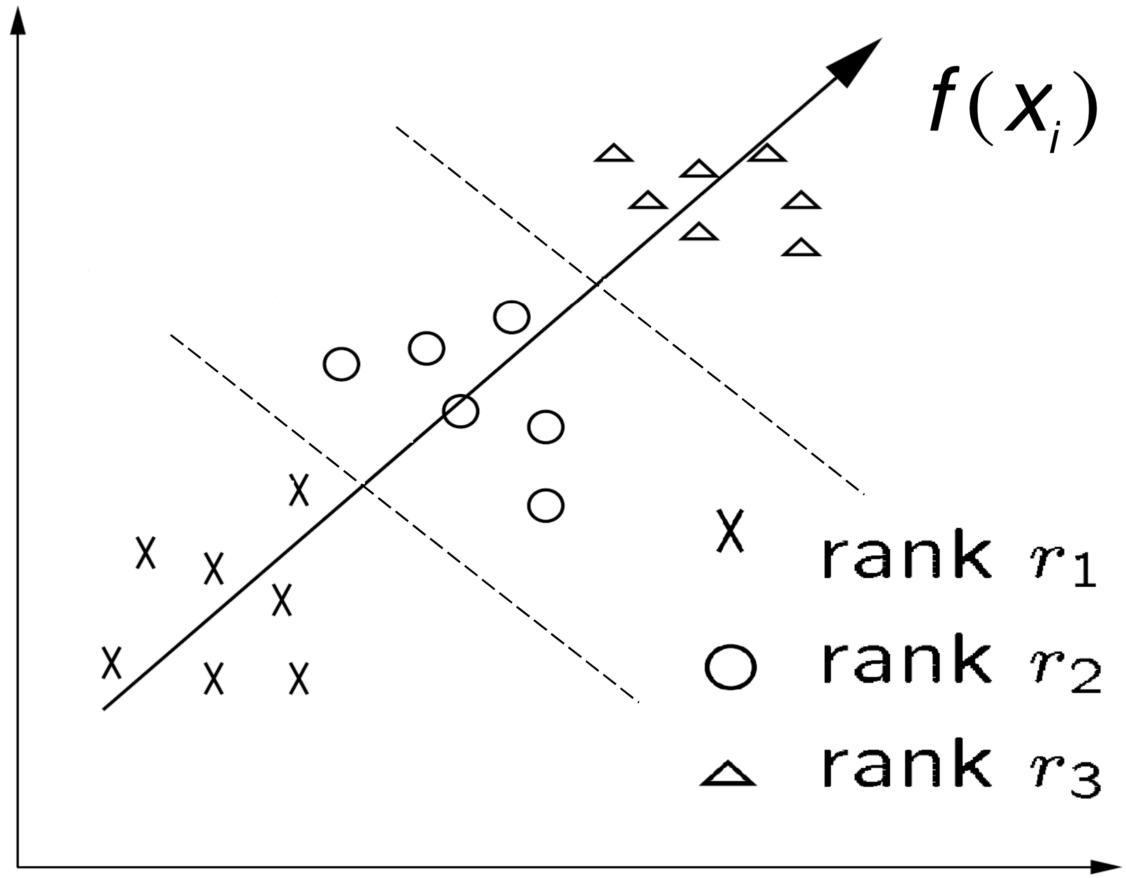
$$\boldsymbol{x}_i > \boldsymbol{x}_j \text{ iff } f(\boldsymbol{x}_i) > f(\boldsymbol{x}_j)$$

- … or at least one that tries to do this with minimal error
- Suppose that f is a linear function

$$f(\boldsymbol{x}_i) = \mathbf{w} \bullet \boldsymbol{x}_i$$

The Ranking SVM

- Ranking Model: $f(\boldsymbol{x}_i)$



The Ranking SVM

- Then (combining the two equations on the last slide):

$$\boldsymbol{x}_i > \boldsymbol{x}_j \text{ iff } \mathbf{w}\bullet\boldsymbol{x}_i - \mathbf{w}\bullet\boldsymbol{x}_j > 0$$

$$\boldsymbol{x}_i > \boldsymbol{x}_j \text{ iff } \mathbf{w}\bullet(\boldsymbol{x}_i - \boldsymbol{x}_j) > 0$$

- Let us then create a new instance space from such pairs: $z_k = x_i - x_k$

$$y_k = +1, -1 \text{ as } \boldsymbol{x}_i \geq, < \boldsymbol{x}_k$$

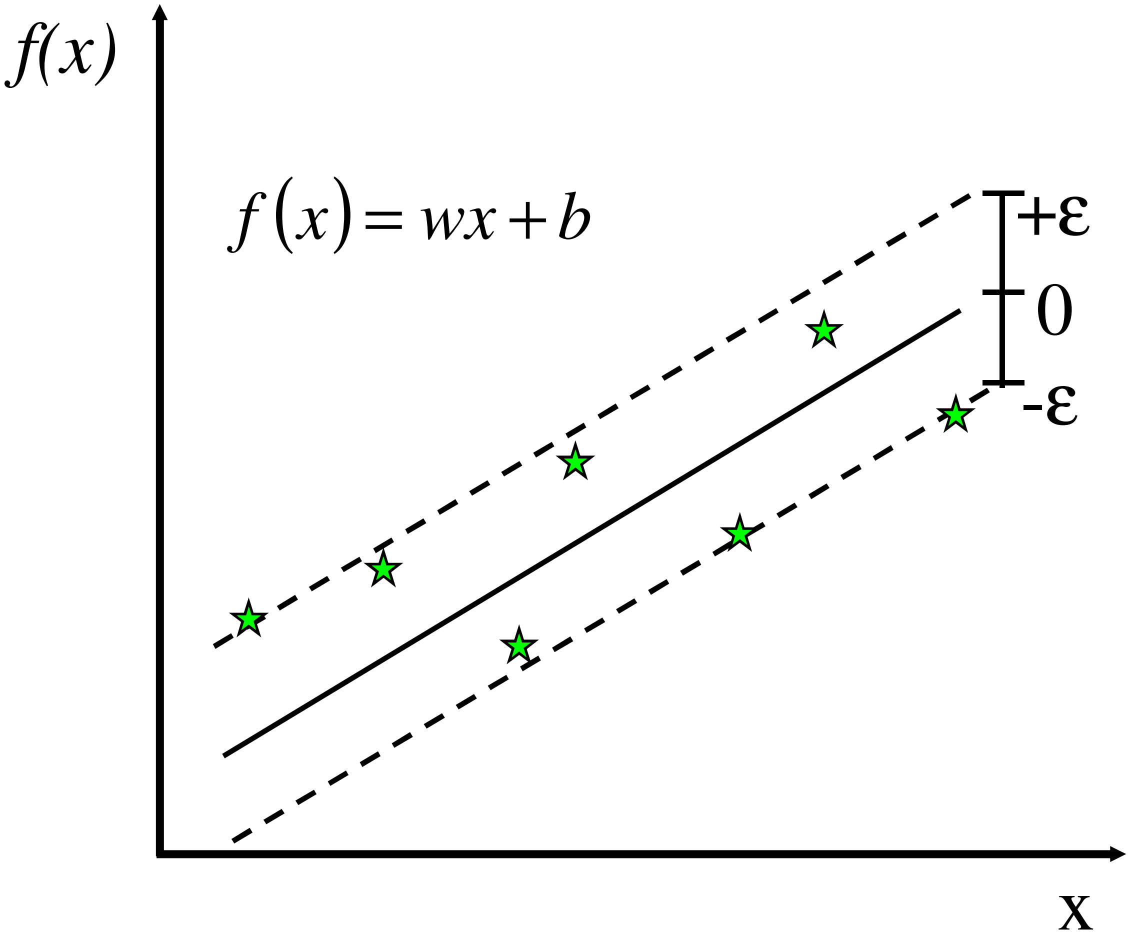
Support Vector Ranking

$$\begin{cases} \min \quad \frac{1}{2}\|\vec{w}\| + C\sum_{i=1}^{m} \xi_i^2 \\ y_k(\vec{w}\cdot(\vec{x_i} - \vec{x_j}) + b) \geq 1 - \xi_k, \quad \forall i, j = 1, .., m \\ \xi_k \geq 0, \quad k = 1, .., m^2 \end{cases}$$

$$y_k = 1 \text{ if } rank(\vec{x_i}) > rank(\vec{x_j}), -1 \text{ otherwise, where } k = i \times m + j$$

- Given two examples we build one example (x_i , x_j)

Support Vector Regression (SVR)



Solution:

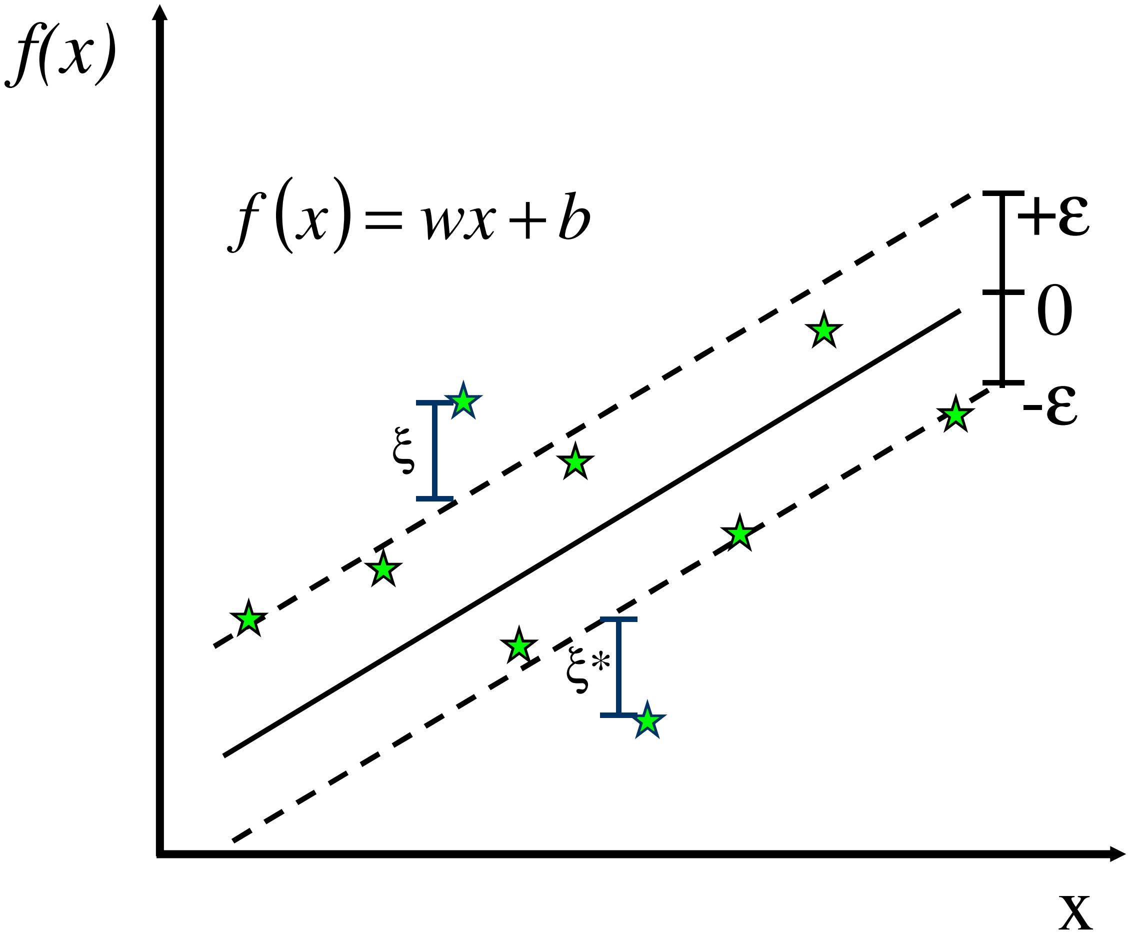
$$Min \frac{1}{2} w^T w$$

Constraints:

$$y_i - w^T x_i - b \leq \varepsilon$$

$$w^T x_i + b - y_i \leq \varepsilon$$

Support Vector Regression (SVR)



Minimise:

$$\frac{1}{2} w^T w + \boxed{C \sum_{i=1}^{N} \left(\xi_i + \xi_i^* \right)}$$

Constraints:

$$y_i - w^T x_i - b \le \varepsilon \boxed{+ \xi_i}$$

$$w^T x_i + b - y_i \le \varepsilon \boxed{+ \xi_i^*}$$

$$\boxed{\xi_i, \xi_i^* \ge 0}$$

Support Vector Regression

$$\min_{\mathbf{w},b,\xi,\xi^*} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$

$$\begin{aligned} \text{s.t. } & y_i - \mathbf{w}^\top \mathbf{x}_i - b \leq \epsilon + \xi_i,\ \ \xi_i \geq 0 \quad \forall 1 \leq i \leq n; \\ & \mathbf{w}^\top \mathbf{x}_i + b - y_i \leq \epsilon + \xi_i^*,\ \ \xi_i^* \geq 0 \quad \forall 1 \leq i \leq n. \end{aligned}$$

- y_i is not -1 or 1 anymore, now it is a value
- ε is the tollerance of our function value

From Binary to Multiclass classifiers

- Three different approaches:
- **ONE-vs-ALL (OVA)**
  - Given the example sets, {E1, E2, E3, …} for the categories: {C1, C2, C3,…} the binary classifiers: {b1, b2, b3,…} are built.
  - For b1, E1 is the set of positives and $E2 \cup E3 \cup \ldots$ is the set of negatives, and so on
  - <u>For testing</u>: given a classification instance x, the category is the one associated with the maximum margin among all binary classifiers

From Binary to Multiclass classifiers

- **ALL-vs-ALL (AVA)**
  - Given the examples: {E1, E2, E3, …} for the categories {C1, C2, C3,…}
    - build the binary classifiers:

      {b1_2, b1_3,…, b1_n, b2_3, b2_4,…, b2_n,…,bn-1_n}
    - by learning on E1 (positives) and E2 (negatives), on E1 (positives) and E3 (negatives) and so on…
  - <u>For testing</u>: given an example x,
    - all the votes of all classifiers are collected
    - where $b_{E1E2} = 1$ means a vote for C1 and $b_{E1E2} = -1$ is a vote for C2
  - Select the category that gets more votes

From Binary to Multiclass classifiers

- **Error Correcting Output Codes** (ECOC)
  - The training set is partitioned according to binary sequences (codes) associated with category sets.
    - For example, 10101 indicates that the set of examples of C1,C3 and C5 are used to train the C_{10101} classifier.
    - The data of the other categories, i.e. C2 and C4 will be negative examples
  - <u>In testing</u>: the code-classifiers are used to decode one the original class, e.g.

$C_{10101} = 1$ and $C_{11010} = 1$ indicates that the instance belongs to C1

That is, the only one consistent with the codes

SVM-light: an implementation of SVMs

- Implements soft margin
- Contains the procedures for solving optimization problems
- Binary classifier
- Examples and descriptions in the web site:

  http://www.joachims.org/

  (http://svmlight.joachims.org/)