
MACHINE LEARNING

Support Vector Machines

Alessandro Moschitti

Department of information and communication technology

University of Trento

Email: moschitti@dit.unitn.it





Summary

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

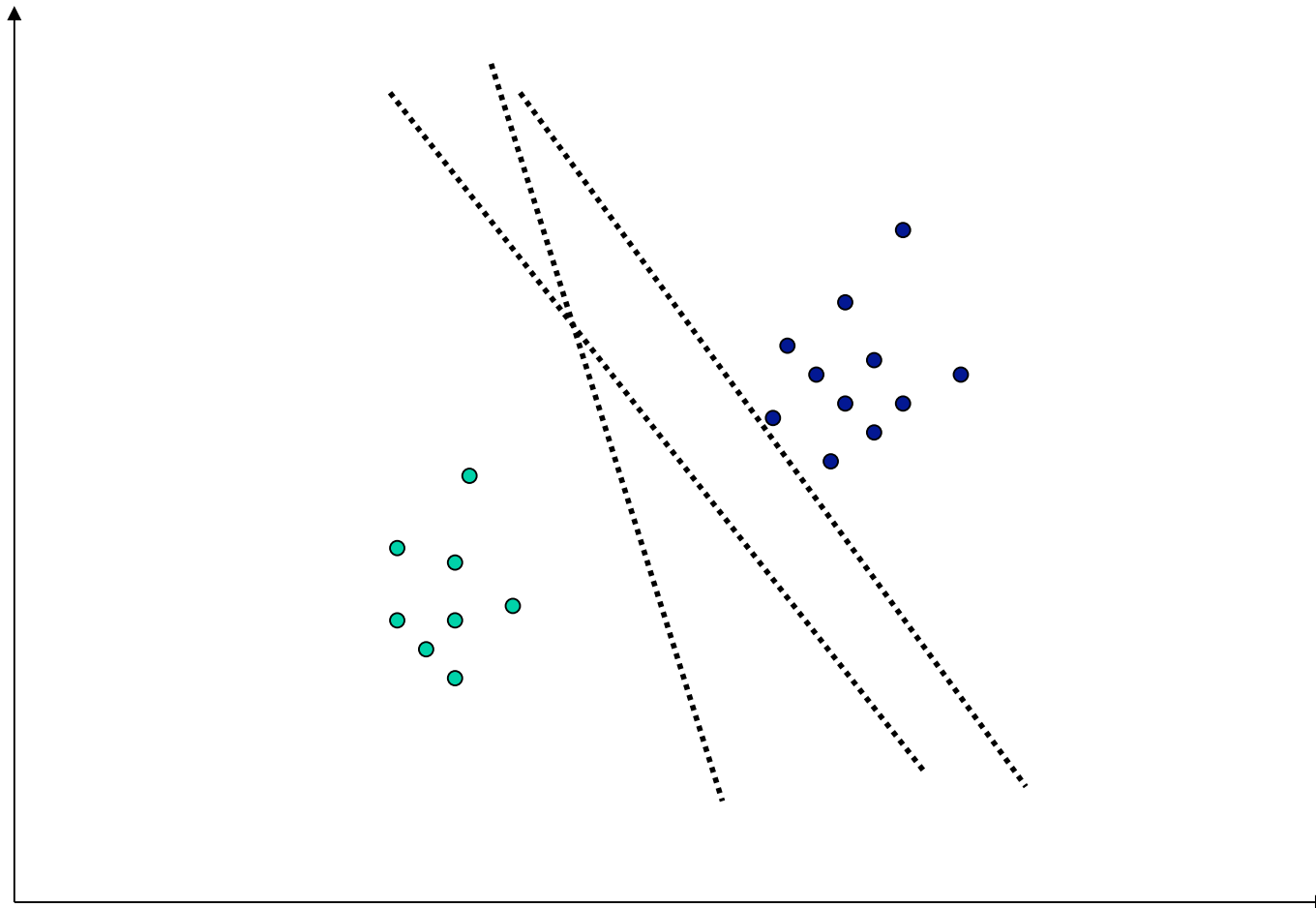


Communications

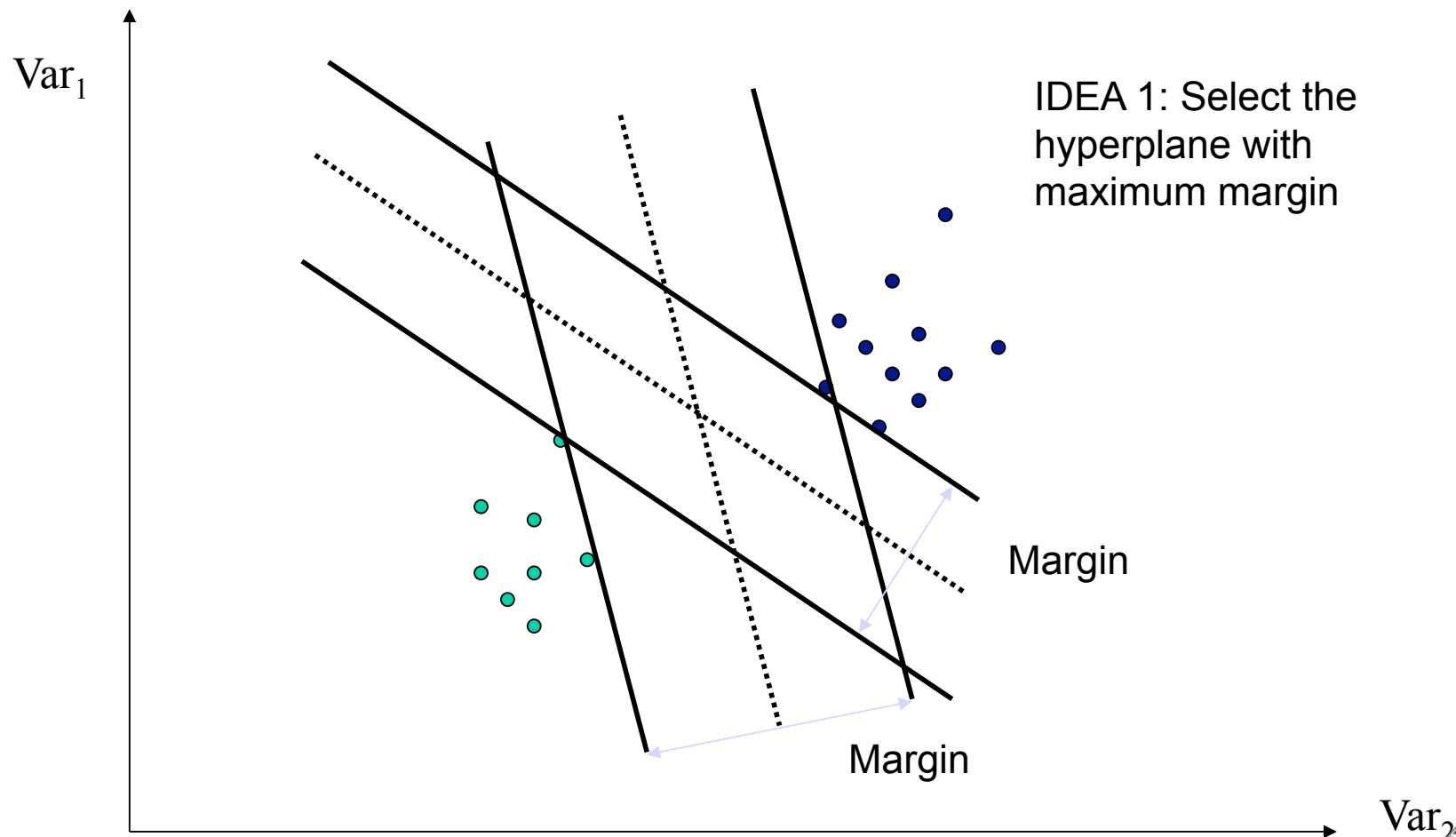
- No lecture tomorrow (neither Dec. 8)
- ML Exams
 - 12 January 2011 at 9:00,
 - 26 January 2011 at 9:00
- Exercise in Lab
 - A201 (Polo scientifico e tecnologico)
 - Wednesday 15 and 22 December, 2011
 - Time: 8.30-10.30



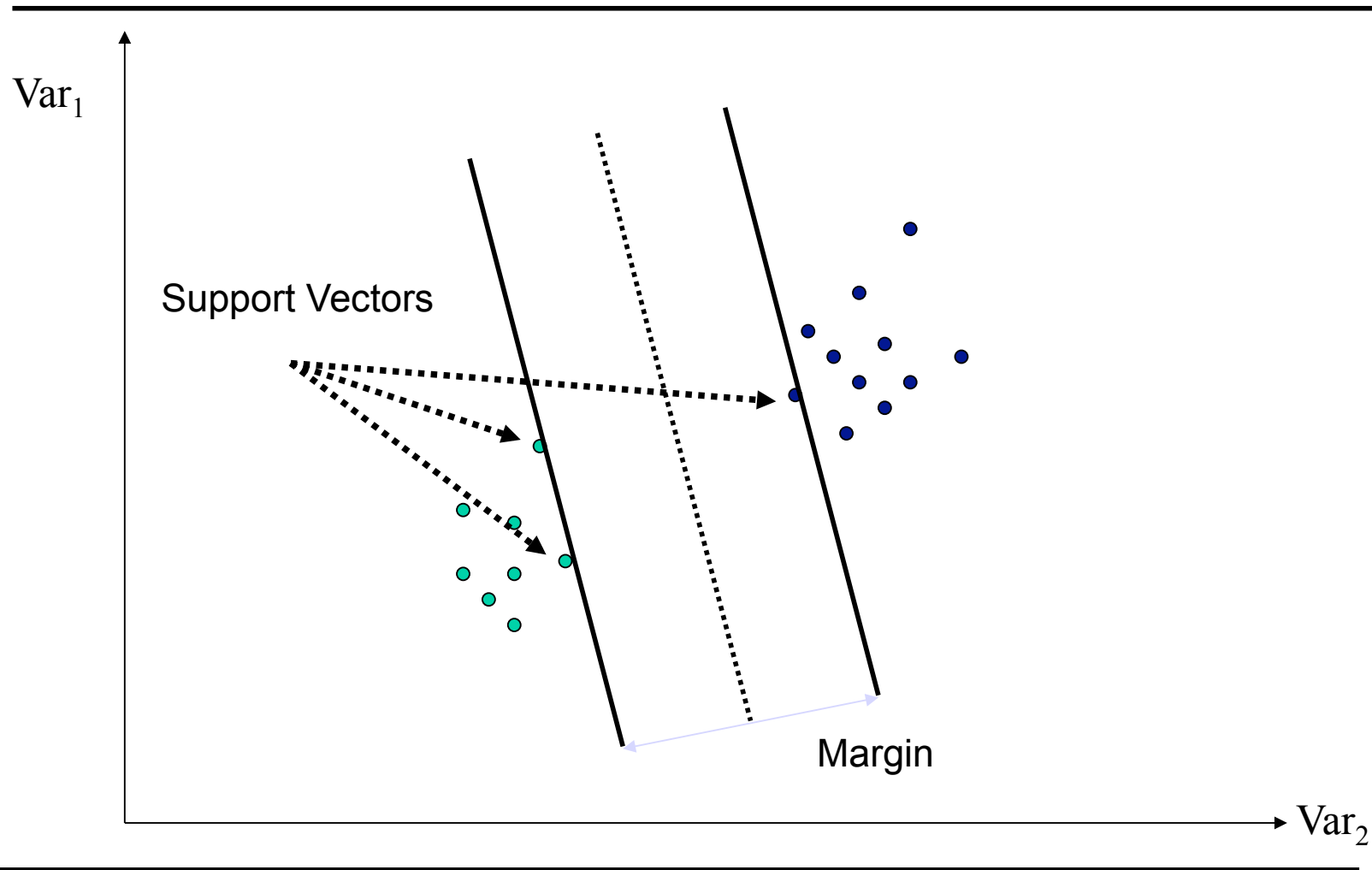
Which hyperplane choose?



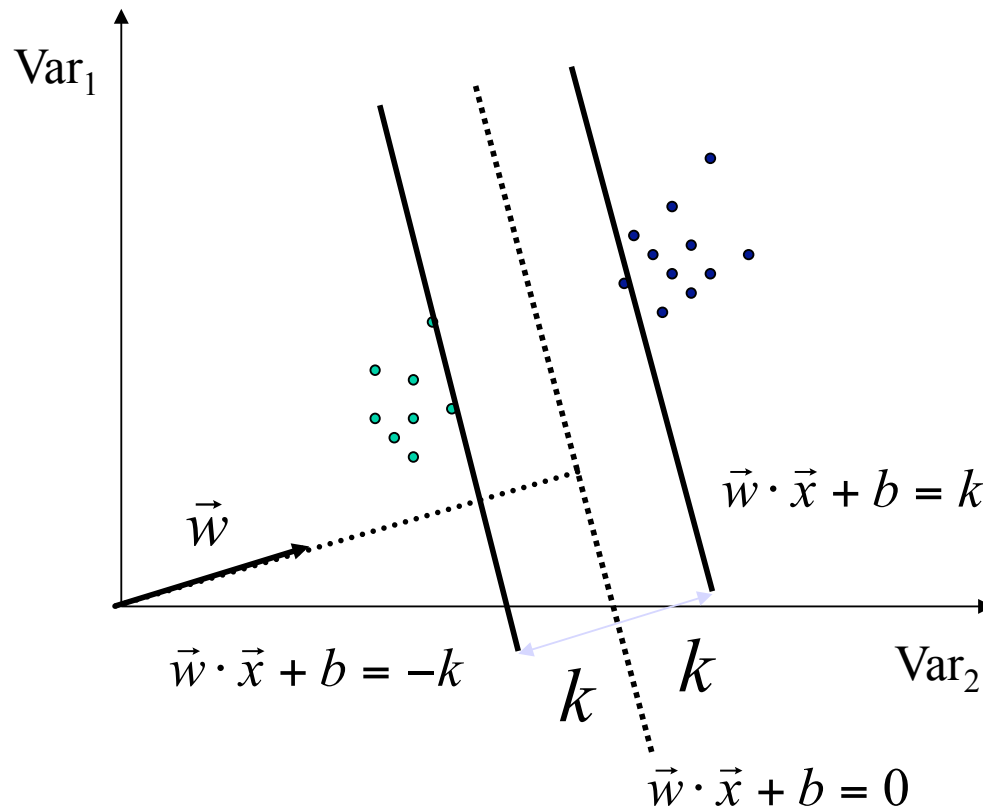
Classifier with a Maximum Margin



Support Vector



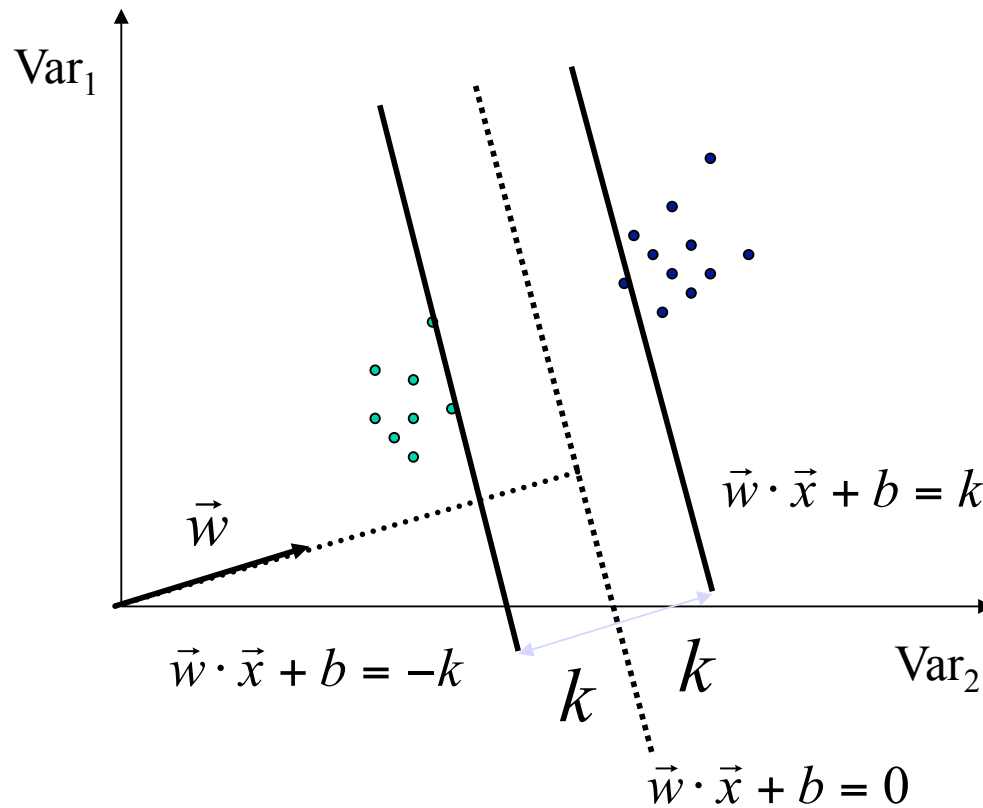
Support Vector Machine Classifiers



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



Support Vector Machines



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

We need to solve

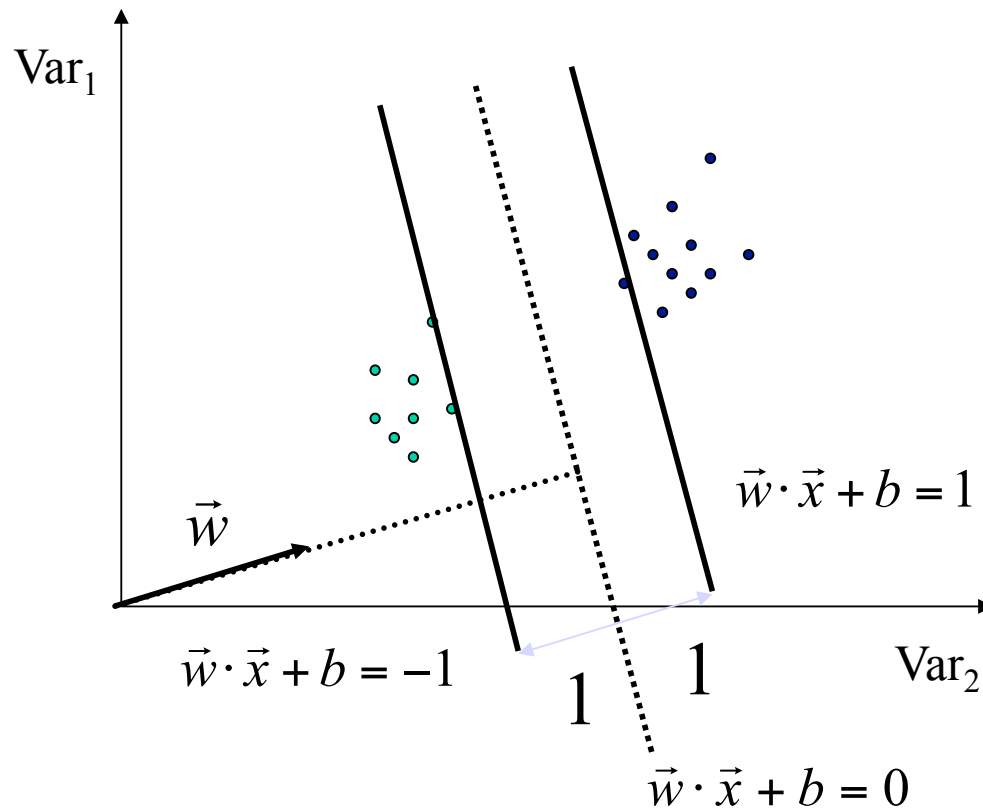
$$\max \frac{2|k|}{\|\vec{w}\|}$$

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

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



Support Vector Machines



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

The problem transforms in:

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

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

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



Final Formulation

$$\begin{aligned} \max \frac{2}{\|\vec{w}\|} \\ \vec{w} \cdot \vec{x}_i + b \geq +1, \quad y_i = 1 \\ \vec{w} \cdot \vec{x}_i + b \leq -1, \quad y_i = -1 \end{aligned} \quad \Rightarrow \quad \begin{aligned} \max \frac{2}{\|\vec{w}\|} \\ y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 \end{aligned} \quad \Rightarrow$$

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



Optimization Problem

- Optimal Hyperplane:

- Minimize $\tau(\vec{w}) = \frac{1}{2} \|\vec{w}\|^2$

- Subject to $y_i ((\vec{w} \cdot \vec{x}_i) + b) \geq 1, i = 1, \dots, m$

- The dual problem is simpler



Lagrangian Definition

Def. 2.24 Let $f(\vec{w})$, $h_i(\vec{w})$ and $g_i(\vec{w})$ be the objective function, the equality constraints and the inequality constraints (i.e. \leq) of an optimization problem, and let $L(\vec{w}, \vec{\alpha}, \vec{\beta})$ be its Lagrangian, defined as follows:

$$L(\vec{w}, \vec{\alpha}, \vec{\beta}) = f(\vec{w}) + \sum_{i=1}^m \alpha_i g_i(\vec{w}) + \sum_{i=1}^l \beta_i h_i(\vec{w})$$



Dual Optimization Problem

The Lagrangian dual problem of the above primal problem is

$$\text{maximize } \theta(\vec{\alpha}, \vec{\beta})$$

$$\text{subject to } \vec{\alpha} \geq \vec{0}$$

$$\text{where } \theta(\vec{\alpha}, \vec{\beta}) = \inf_{w \in W} L(\vec{w}, \vec{\alpha}, \vec{\beta})$$



Dual Transformation

- Given the Lagrangian associated with our problem

$$L(\vec{w}, b, \vec{\alpha}) = \frac{1}{2} \vec{w} \cdot \vec{w} - \sum_{i=1}^m \alpha_i [y_i (\vec{w} \cdot \vec{x}_i + b) - 1]$$

- To solve the dual problem we need to evaluate:

$$\theta(\vec{\alpha}, \vec{\beta}) = \inf_{\vec{w} \in W} L(\vec{w}, \vec{\alpha}, \vec{\beta})$$

- Let us impose the derivatives to 0, with respect to \vec{w}

$$\frac{\partial L(\vec{w}, b, \vec{\alpha})}{\partial \vec{w}} = \vec{w} - \sum_{i=1}^m y_i \alpha_i \vec{x}_i = \vec{0} \quad \Rightarrow \quad \vec{w} = \sum_{i=1}^m y_i \alpha_i \vec{x}_i$$



Dual Transformation (cont'd)

- and wrt b

$$\frac{\partial L(\vec{w}, b, \vec{\alpha})}{\partial b} = \sum_{i=1}^m y_i \alpha_i = 0$$

- Then we substituted them in the Lagrange function

$$\begin{aligned} L(\vec{w}, b, \vec{\alpha}) &= \frac{1}{2} \vec{w} \cdot \vec{w} - \sum_{i=1}^m \alpha_i [y_i (\vec{w} \cdot \vec{x}_i + b) - 1] = \\ &= \frac{1}{2} \sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j - \sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j + \sum_{i=1}^m \alpha_i \\ &= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j \end{aligned}$$



Final Dual Problem

$$\text{maximize} \quad \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j$$

$$\text{subject to} \quad \alpha_i \geq 0, \quad i = 1, \dots, m$$

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



Khun-Tucker Theorem

- Necessary and sufficient conditions to optimality

$$\frac{\partial L(\vec{w}^*, \vec{\alpha}^*, \vec{\beta}^*)}{\partial \vec{w}} = \vec{0}$$

$$\frac{\partial L(\vec{w}^*, \vec{\alpha}^*, \vec{\beta}^*)}{\partial b} = \vec{0}$$

$$\alpha_i^* g_i(\vec{w}^*) = 0, \quad i = 1, \dots, m$$

$$g_i(\vec{w}^*) \leq 0, \quad i = 1, \dots, m$$

$$\alpha_i^* \geq 0, \quad i = 1, \dots, m$$



Properties coming from constraints

■ Lagrange constraints: $\sum_{i=1}^m a_i y_i = 0 \quad \vec{w} = \sum_{i=1}^m \alpha_i y_i \vec{x}_i$

- Karush-Kuhn-Tucker constraints

$$\alpha_i \cdot [y_i (\vec{x}_i \cdot \vec{w} + b) - 1] = 0, \quad i = 1, \dots, m$$

- Support Vectors have α_i not null

- To evaluate b , we can apply the following equation

$$b^* = -\frac{\vec{w}^* \cdot \vec{x}^+ + \vec{w}^* \cdot \vec{x}^-}{2}$$



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

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

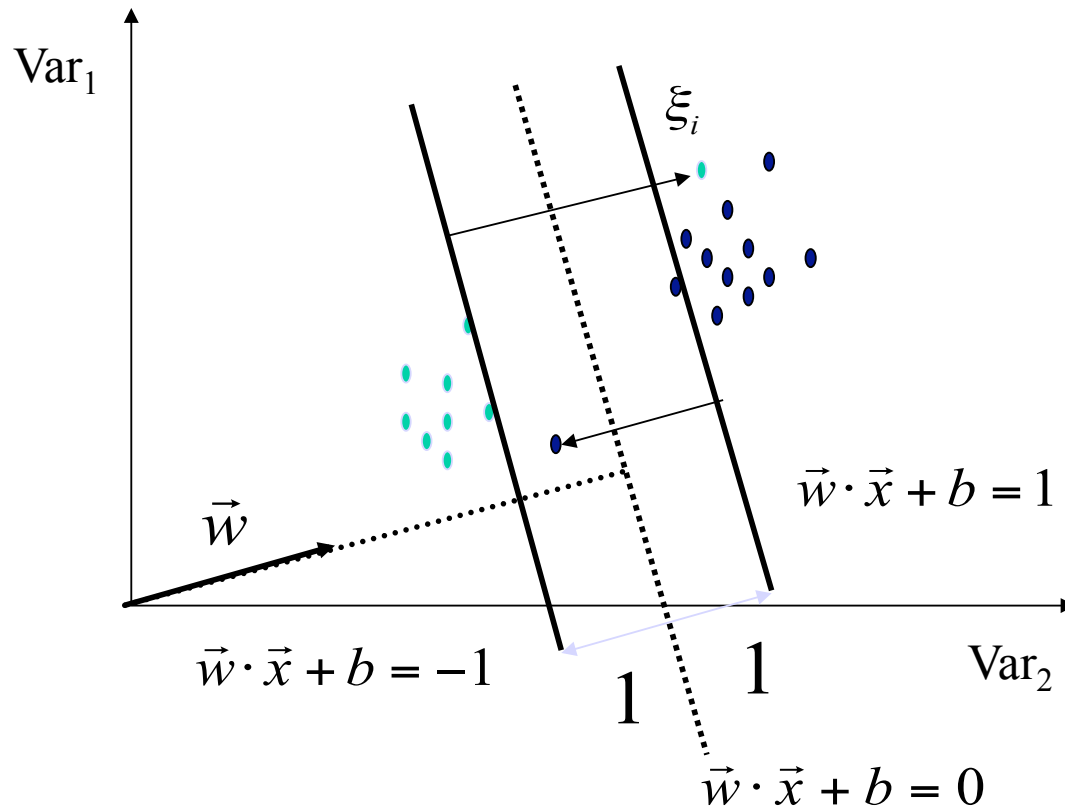
$$(x, y) \cdot (1/\sqrt{2}, 1/\sqrt{2}) + b = 0 \Rightarrow x/\sqrt{2} + y/\sqrt{2} = -b$$

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

- 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

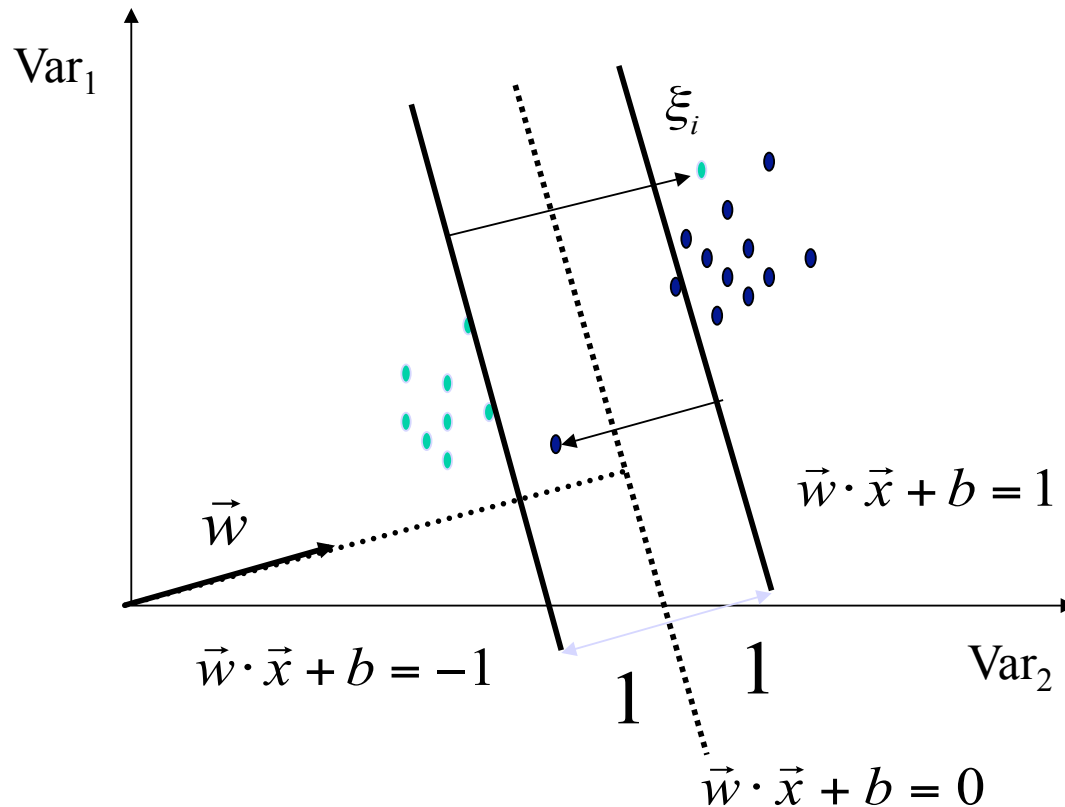


ξ_i slack variables are added

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



Dual formulation

$$\begin{cases} \min & \frac{1}{2} \|\vec{w}\|^2 + C \sum_{i=1}^m \xi_i^2 \\ & y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i, \quad \forall i = 1, \dots, m \\ & \xi_i \geq 0, \quad i = 1, \dots, m \end{cases}$$

$$L(\vec{w}, b, \vec{\xi}, \vec{\alpha}) = \frac{1}{2} \vec{w} \cdot \vec{w} + \frac{C}{2} \sum_{i=1}^m \xi_i^2 - \sum_{i=1}^m \alpha_i [y_i(\vec{w} \cdot \vec{x}_i + b) - 1 + \xi_i],$$

- By deriving wrt $\vec{w}, \vec{\xi}$ and b



Partial Derivatives

$$\frac{\partial L(\vec{w}, b, \vec{\xi}, \vec{\alpha})}{\partial \vec{w}} = \vec{w} - \sum_{i=1}^m y_i \alpha_i \vec{x}_i = \vec{0} \quad \Rightarrow \quad \vec{w} = \sum_{i=1}^m y_i \alpha_i \vec{x}_i$$

$$\frac{\partial L(\vec{w}, b, \vec{\xi}, \vec{\alpha})}{\partial \vec{\xi}} = C \vec{\xi} - \vec{\alpha} = \vec{0}$$

$$\frac{\partial L(\vec{w}, b, \vec{\xi}, \vec{\alpha})}{\partial b} = \sum_{i=1}^m y_i \alpha_i = 0$$



Substitution in the objective function

$$\begin{aligned} &= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j + \frac{1}{2C} \vec{a} \cdot \vec{a} - \frac{1}{C} \vec{a} \cdot \vec{a} = \\ &= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j - \frac{1}{2C} \vec{a} \cdot \vec{a} = \\ &= \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j \left(\vec{x}_i \cdot \vec{x}_j + \frac{1}{C} \delta_{ij} \right), \end{aligned}$$

- δ_{ij} of Kronecker



Final dual optimization problem

$$\sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y_i y_j \alpha_i \alpha_j (\vec{x}_i \cdot \vec{x}_j + \frac{1}{C} \delta_{ij})$$

$$\alpha_i \geq 0, \quad \forall i = 1, \dots, m$$

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



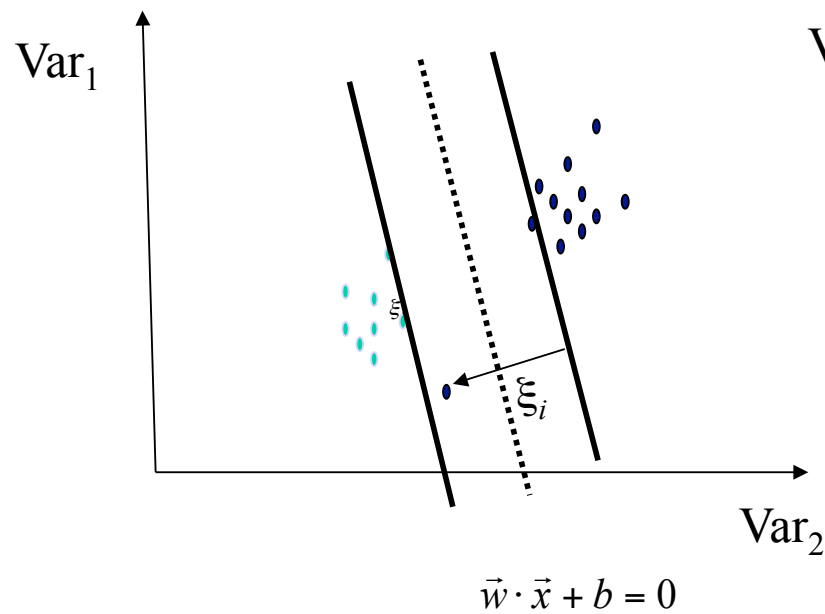
Soft Margin Support Vector Machines

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

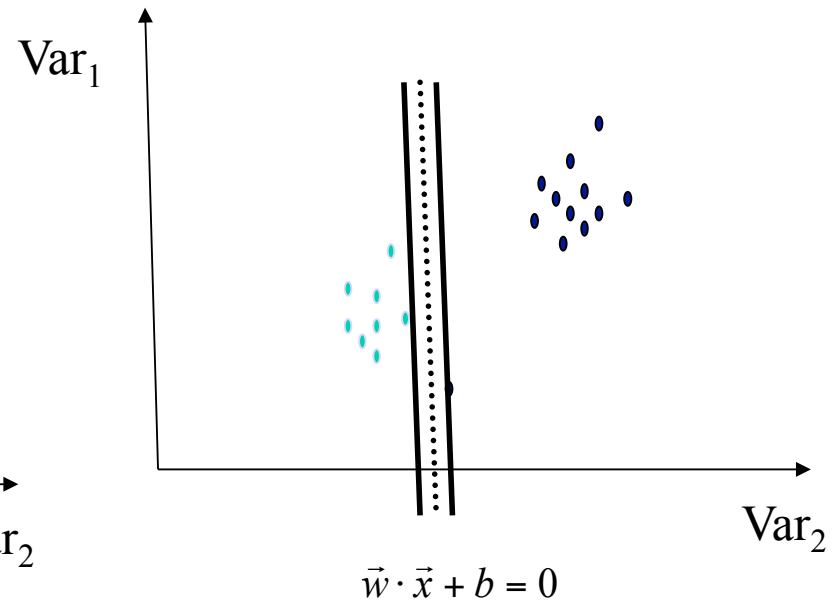
- The algorithm tries to keep ξ_i low and maximize the margin
- NB: The number of error is not directly minimized (NP-complete problem); the distances from the hyperplane are minimized
- If $C \rightarrow \infty$, the solution tends to the one of the *hard-margin* algorithm
- *Attention !!!*: if $C = 0$ 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



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



Soft Margin SVM



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

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

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



Theoretical Justification



Definition of Training Set error

- Training Data

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

- Empirical Risk (error)

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

- Risk (error)

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



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 the VC-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

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

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



Error Characterization (part 2)

Lemma 1. [Vapnik, 1982] Consider hyperplanes $h(\vec{d}) = \text{sign}\{\vec{w} \cdot \vec{d} + b\}$ 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

$$|\vec{w} \cdot \vec{d}_i + b| \geq 1, \text{ with } \|\vec{w}\| = A \quad (5)$$

then this set of hyperplane has a VCdim d bounded by

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



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

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

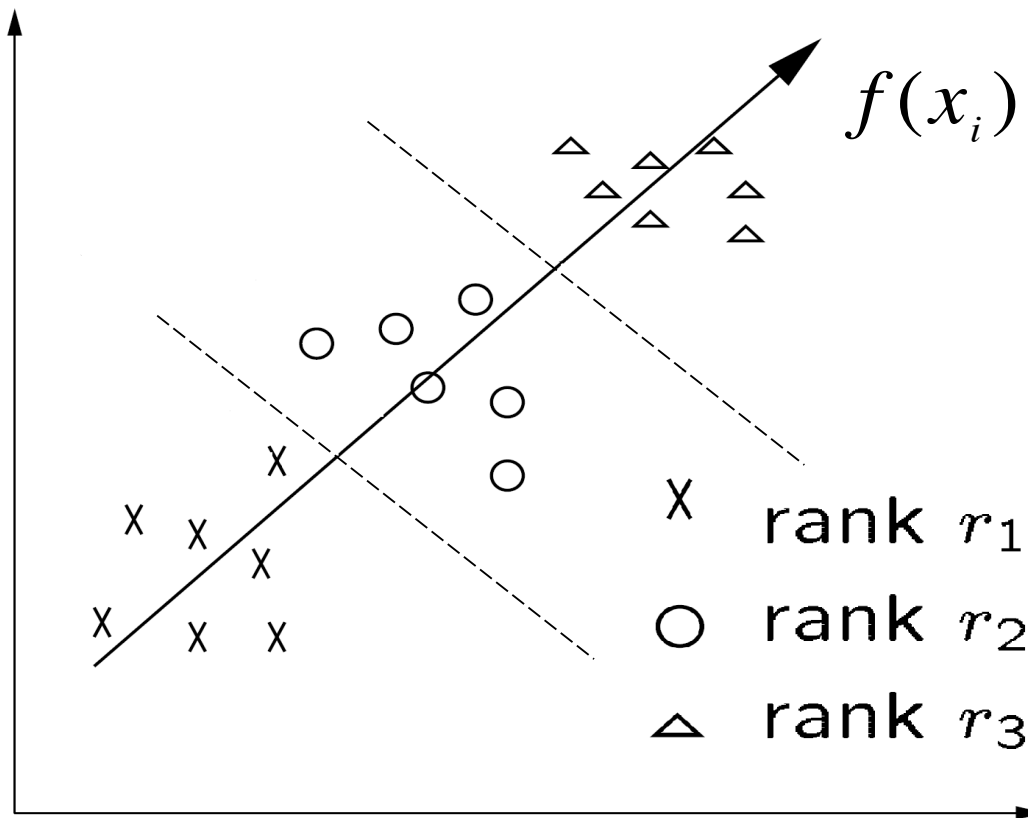
- ... or at least one that tries to do this with minimal error
- Suppose that f is a linear function

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



The Ranking SVM

- Ranking Model: $f(x_i)$



The Ranking SVM

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

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

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

- Let us then create a new instance space from such pairs:

$$z_k = \mathbf{x}_i - \mathbf{x}_k$$

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



Support Vector Ranking

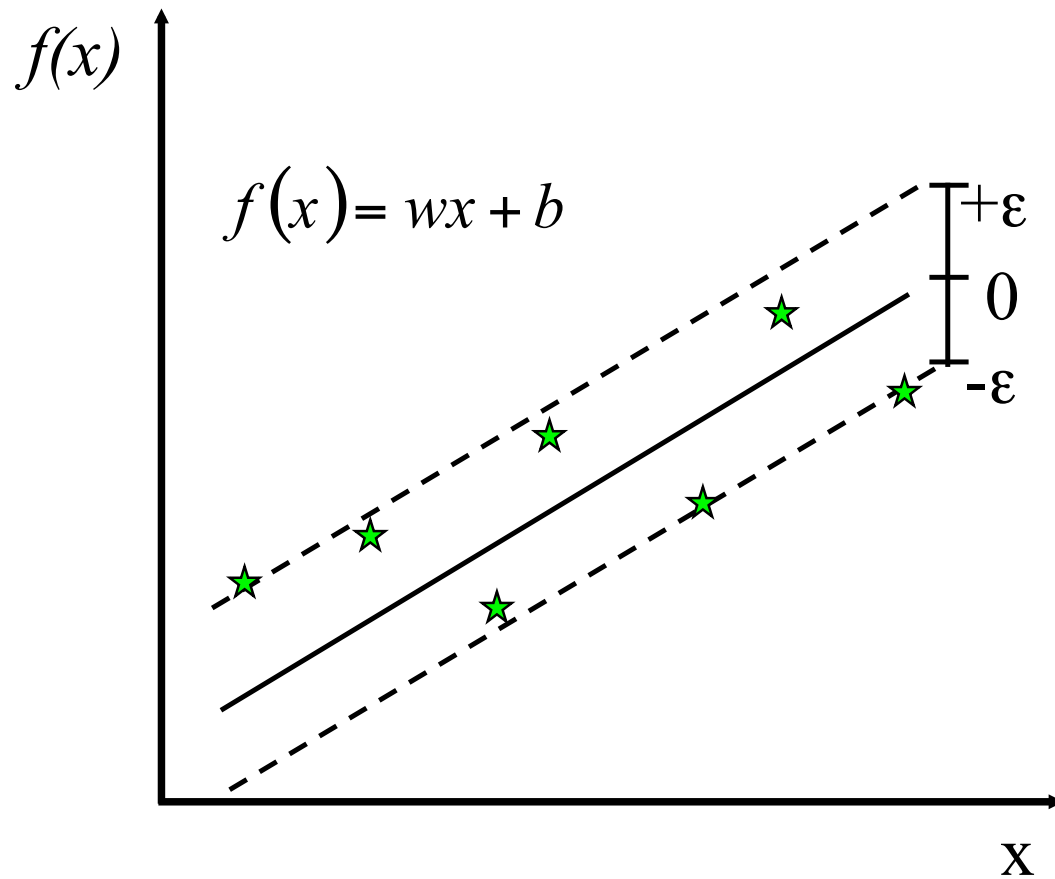
$$\begin{cases} \min & \frac{1}{2} \|\vec{w}\|^2 + C \sum_{i=1}^m \xi_i^2 \\ & y_k (\vec{w} \cdot (\vec{x}_i - \vec{x}_j) + b) \geq 1 - \xi_k, \quad \forall i, j = 1, \dots, m \\ & \xi_k \geq 0, \quad k = 1, \dots, m^2 \end{cases}$$

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

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



Support Vector Regression (SVR)



Solution:

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

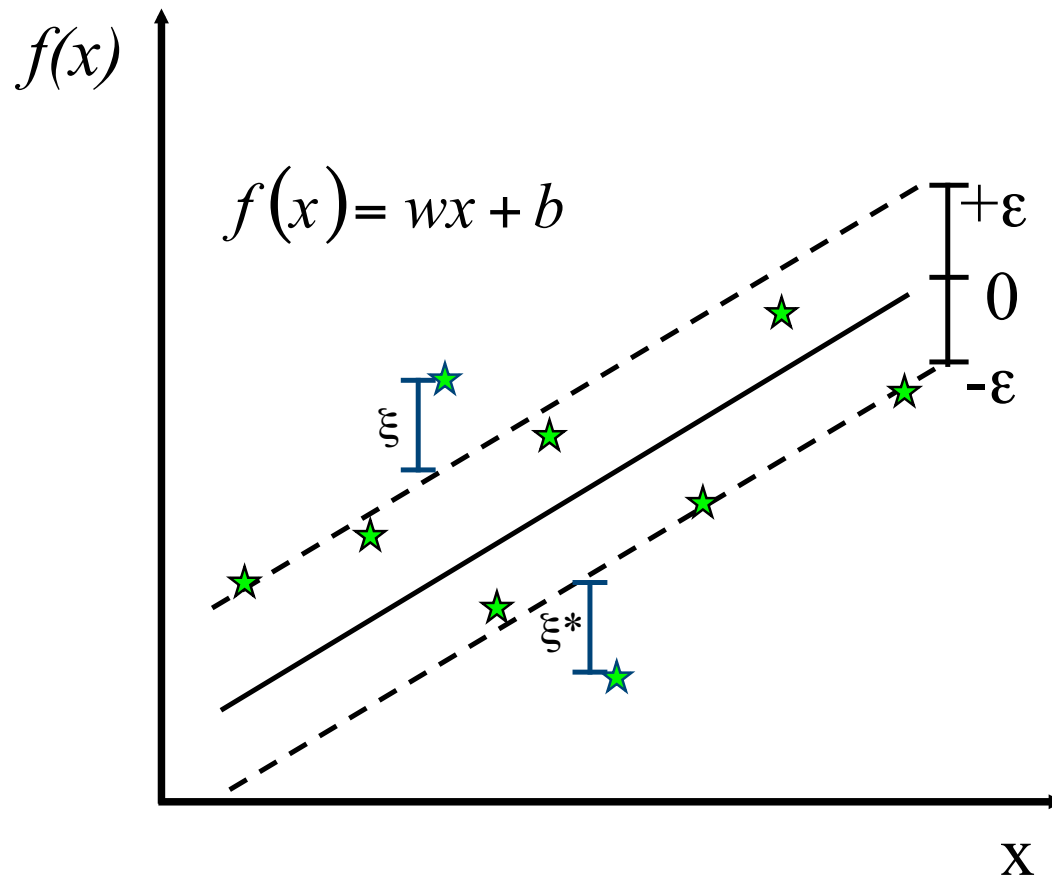
Constraints:

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

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



Support Vector Regression (SVR)



Minimise:

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

Constraints:

$$y_i - w^T x_i - b \leq \epsilon + \xi_i$$

$$w^T x_i + b - y_i \leq \epsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \geq 0$$



Support Vector Regression

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

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

$$\mathbf{w}^\top \mathbf{x}_i + b - y_i \leq \epsilon + \xi_i^*, \quad \xi_i^* \geq 0 \quad \forall 1 \leq i \leq n.$$

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



From Binary to Multiclass classifiers

- Three different approaches:
- **ONE-vs-ALL (OVA)**
 - Given the example sets, $\{E_1, E_2, E_3, \dots\}$ for the categories: $\{C_1, C_2, C_3, \dots\}$ the binary classifiers: $\{b_1, b_2, b_3, \dots\}$ are built.
 - For b_1 , E_1 is the set of positives and $E_2 \cup E_3 \cup \dots$ is the set of negatives, and so on
 - For testing: 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: $\{E1, E2, E3, \dots\}$ for the categories $\{C1, C2, C3, \dots\}$
 - build the binary classifiers:
 $\{b1_2, b1_3, \dots, b1_n, b2_3, b2_4, \dots, b2_n, \dots, bn-1_n\}$
 - by learning on E1 (positives) and E2 (negatives), on E1 (positives) and E3 (negatives) and so on...
- For testing: 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
- In testing: 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/>)



References

- *A tutorial on Support Vector Machines for Pattern Recognition*
 - **Downloadable article (Chriss Burges)**
- *The Vapnik-Chervonenkis Dimension and the Learning Capability of Neural Nets*
 - **Downloadable Presentation**
- Computational Learning Theory
(Sally A Goldman Washington University St. Louis Missouri)
 - **Downloadable Article**
- *AN INTRODUCTION TO SUPPORT VECTOR MACHINES*
(and other kernel-based learning methods)
N. Cristianini and J. Shawe-Taylor Cambridge University Press
 - **Check our library**
- *The Nature of Statistical Learning Theory*
Vladimir Naumovich Vapnik - Springer Verlag (December, 1999)
 - **Check our library**



Exercise

- 1. The equations of SVMs for Classification, Ranking and Regression (you can get them from my slides).
- 2. The perceptron algorithm for Classification, Ranking and Regression (the last two you have to provide by looking at what you wrote in point (1)).
- 3. The same as point (2) by using kernels (write the kernel definition as introduction of this section).

