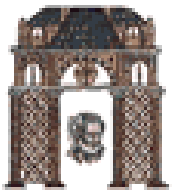
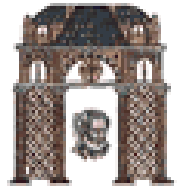


Αναγνώριση Προτύπων

Σύστημα Διανυσμάτων Υποστήριξης (Support Vector Machine)

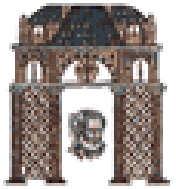
Χριστόδουλος Χαμζάς

Τα περιεχόμενα των παρουσιάσεων προέρχονται από τις παρουσιάσεις του αντίστοιχου διδακτέου μαθήματος του καθ. Παναγιώτη Τσακαλίδη, Τμ. Επιστήμης Υπολογιστών, Παν. Κρήτης και του καθ. Σέργιου Θεοδωρίδη, Τμήμα Πληροφορικής, Πανεπιστήμιο Αθηνών. Βασίζεται στα βιβλία: "Pattern Classification", R.O. Duda, P.E. Hart, D.G. Stork, Wiley, 2nd Ed., 2001 και S. Theodoridis, K. Koutroumbas, Pattern Recognition, 3rd Edition, Academic Press, 2006

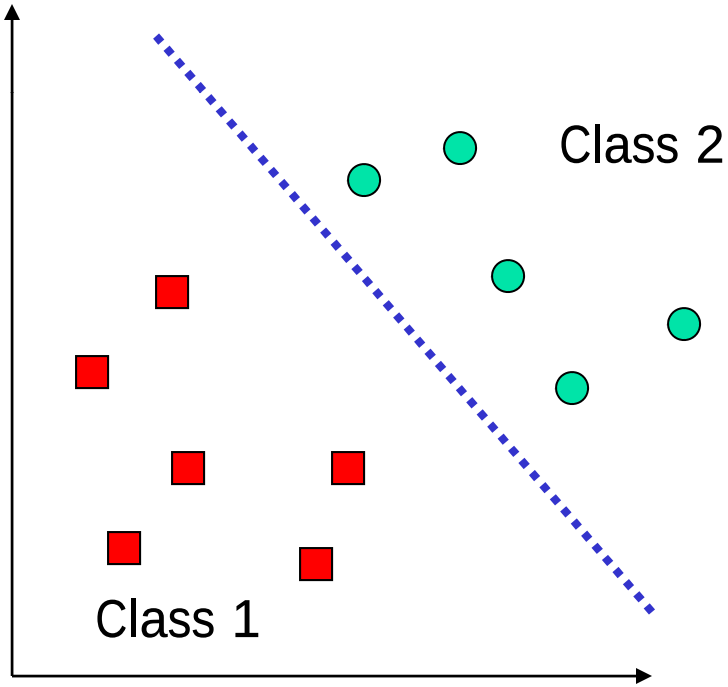


SUPPORT VECTOR MACHINE

- Έως τώρα είδαμε ότι στο πρόβλημα με μόνο 2 κατηγορίες οι οποίες είναι γραμμικά διαχωρίσιμες μπορούμε να έχουμε πολλούς γραμμικούς ταξινομητές που να χωρίζουν τα δύο σύνολα.

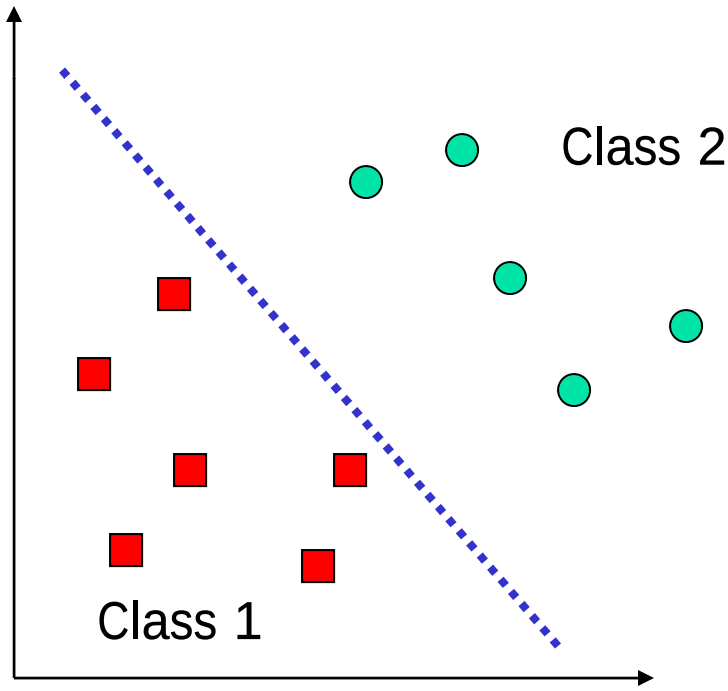


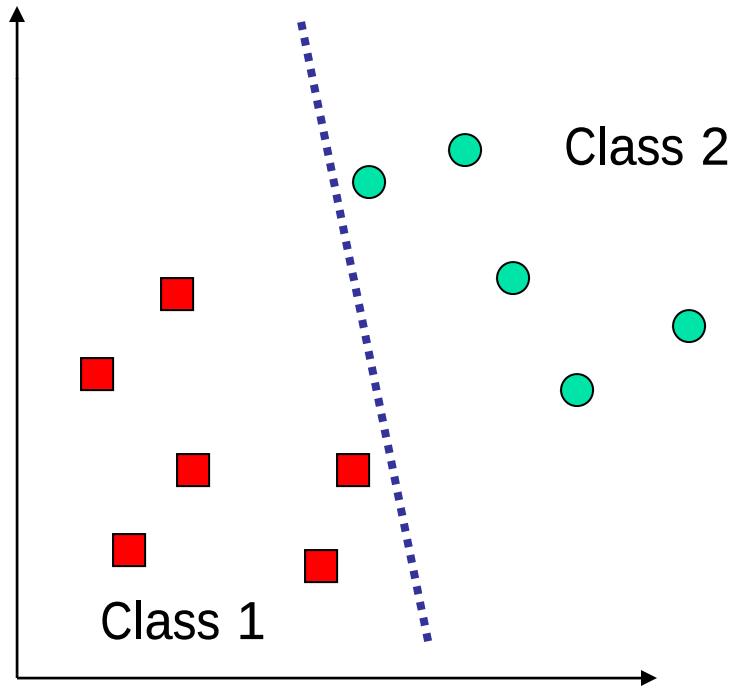
Πρόβλημα με 2 κατηγορίες: Περίπτωση γραμμικά διαχωρίσιμων κατηγοριών

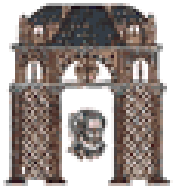


- Υπάρχουν πολλοί κατάλληλοι γραμμικοί ταξινομητές
- Ποιόν να διαλέξουμε? Ποιος είναι ο καλύτερος?

Παράδειγμα κακών ταξινομητών

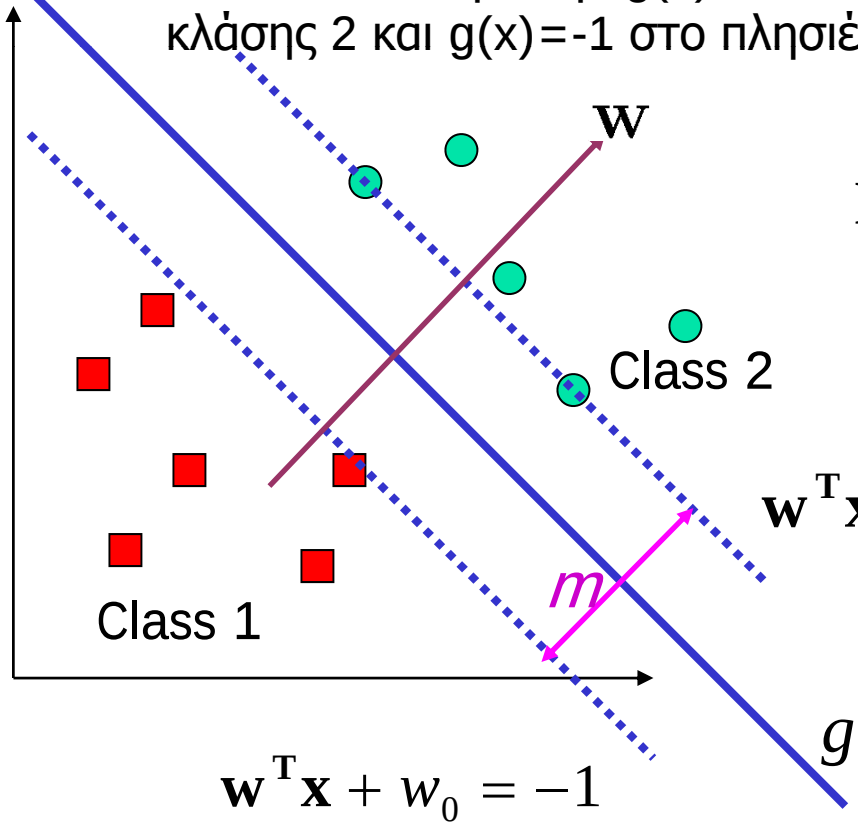






Καλός ταξινομητής: Το σύνορο πρέπει να απέχει πολύ από τα δεδομένα

- Το σύνορο που διαχωρίζει τις κατηγορίες, πρέπει να είναι όσο το δυνατόν πιο μακρυά από τα δεδομένα
  - Πρέπει συνεπώς να μεγιστοποιήσουμε το περιθώριο (margin), *m*
  - Κανονικοποιούμε την g(x) έτσι ώστε g(x)=1 στο πλησιέστερο σημείο της κλάσης 2 και g(x)=-1 στο πλησιέστερο σημείο της κλάσης 1



$$\text{margin } m = \frac{2}{\|\mathbf{w}\|}$$

$$\mathbf{w}^T\mathbf{x} + w_0 = 1$$

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = 0$$

$$\mathbf{w}^T\mathbf{x} + w_0 = -1$$

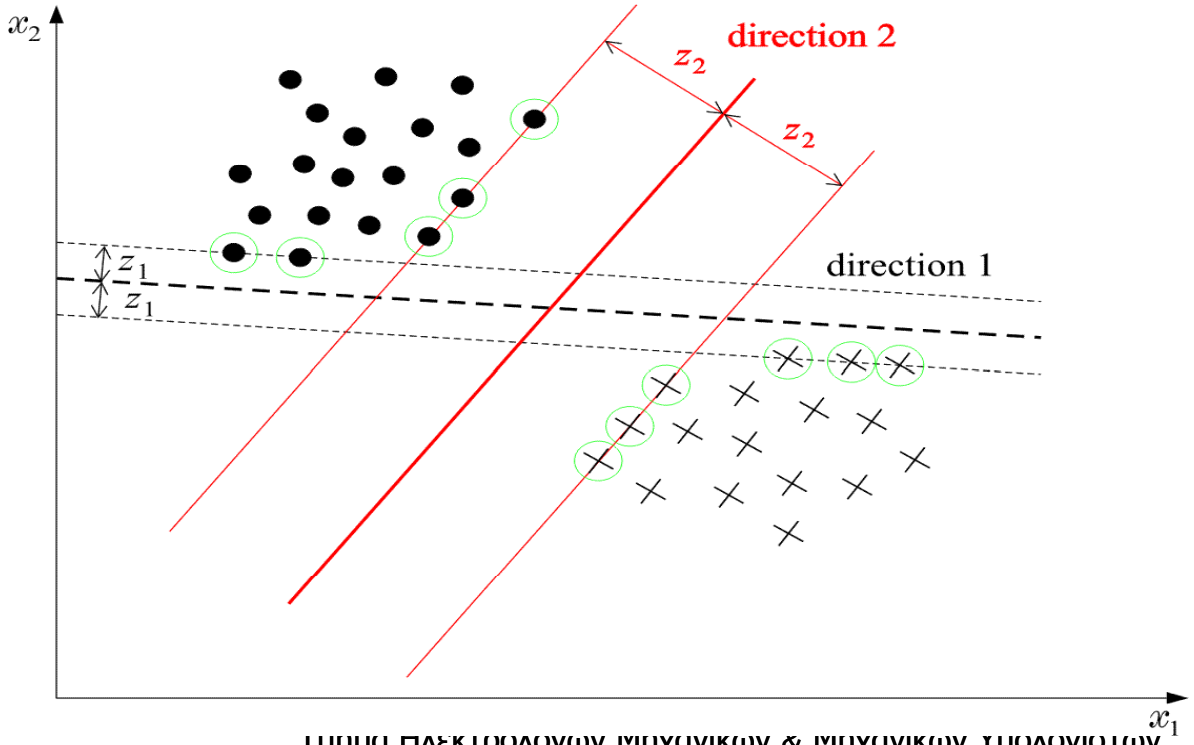
SUPPORT VECTOR MACHINES

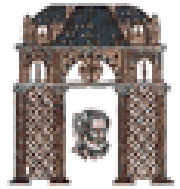
(Μηχανές Διανυσματικής Στήριξης)

- Στόχος: Εάν έχουμε, δύο κατηγορίες που είναι γραμμικά διαχωρίσιμες , να βρείς την συνάρτηση διάκρισης

$$g(\underline{x}) = \underline{w}^T \underline{x} + w_0 = 0$$

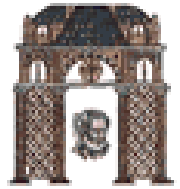
- η οποία αφήνει την μέγιστη απόσταση, περιθώριο, (maximum margin) και από τις δύο κατηγορίες (from both classes)





Margin (Περιθώριο) : Κάθε υπερεπίπεδο $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \mathbf{w}_0$ χαρακτηρίζεται από

- Την κατεύθυνση (κλίση) στο χώρο δηλ. $\mathbf{w}$
- Την θέση του στον χώρο, δηλ. $\mathbf{w}_0$
- Για ΚΑΘΕ κατεύθυνση διάλεξε εκείνο το υπερεπίπεδο το οποίο απέχει την ΙΔΙΑ απόσταση από τα πλησιέστερα σημεία κάθε κλάσης.
- Το περιθώριο (margin) είναι διπλάσιο αυτής της απόστασης.



- Η απόσταση ενός σημείου από το υπερεπίπεδο δίνεται από

$$z_{\hat{x}} = \frac{g(\underline{\hat{x}})}{\|\mathbf{w}\|}$$

- Κανονικοποίησε τα, ***w*** και ***w₀***, έτσι ώστε στα πλησιέστερα δεδομένα και από τις δύο κατηγορίες, η συνάρτηση διαχωρισμού να λαμβάνει τις τιμές ±1:

$$|g(\underline{\mathbf{x}})| = 1 \ \{g(\mathbf{x}) = +1 \text{ for } \omega_1 \text{ and } g(\mathbf{x}) = -1 \text{ for } \omega_2\}$$

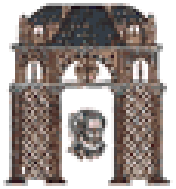
- Συνεπώς το περιθώριο (margin) δίνεται από

$$m = \frac{1}{\|\mathbf{w}\|} + \frac{1}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$

- Επίσης ισχύει ότι

$$\mathbf{w}^{\mathbf{T}}\mathbf{x} + w_0 \geq 1 \ \ \forall \mathbf{x} \in \omega_1$$

$$\mathbf{w}^{\mathbf{T}}\mathbf{x} + w_0 \leq -1 \ \ \forall \mathbf{x} \in \omega_2$$



Το μαθηματικό πρόβλημα

- SVM (γραμμικός) ταξινομητής

$$g(\underline{x}) = \underline{w}^T \underline{x} + w_0$$

- Ελαχιστοποίησε

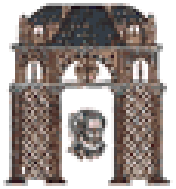
$$J(\underline{w}) = \frac{1}{2}\|\underline{w}\|^2$$

- Με τον περιορισμό

$$y_i(\underline{w}^T \underline{x}_i + w_0) \geq 1, \; i = 1,2,\ldots,N$$

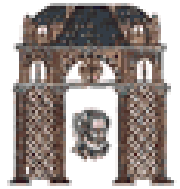
$$y_i = 1, \text{for } \underline{x}_i \in \omega_i,$$
$$y_i = -1, \text{for } \underline{x}_i \in \omega_2$$

- Αυτό ισχύει επειδή ελαχιστοποιώντας το ||w|| το περιθώριο $\frac{2}{\|w\|}$ γίνεται μέγιστο



- Το προηγούμενο είναι ένα πρόβλημα quadratic optimization task υποκείμενο σε ένα σύνολο γραμμικών ανισοτητικών περιορισμών. Οι συνθήκες Karush-Kuhh-Tucker, δηλώνουν ότι η βέλτιστη λύση (the minimizer) ικανοποιεί :
  - (1) $\frac{\partial}{\partial \underline{w}} \mathrm{L}(\underline{w}, w_0, \underline{\alpha}) = 0$
  - (2) $\frac{\partial}{\partial w_0} L(\underline{w}, w_0, \underline{\alpha}) = 0$
  - (3) $\alpha_i \geq 0, i = 1,2,..., N$
  - (4) $\alpha_i \left[y_i (\mathbf{w}^T \mathbf{x}_i + w_0) - 1 \right] = 0, i = 1,2,..., N$
  - Όπου $L(.,.,.)$ είναι η Lagrangian

$$L(\mathbf{w}, w_0, \mathbf{a}) \equiv \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^{N} \alpha_i [y_i (\mathbf{w}^T \mathbf{x}_i + w_0)]$$

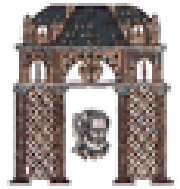


Η λύση: από τα προηγούμενα προκύπτει

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

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

$$a_i \geq 0, \quad y_i = \begin{cases} 1 \text{ if } \mathbf{x_i} \in \omega_1 \\ -1 \text{ if } \mathbf{x_i} \in \omega_2 \end{cases}$$



Σχόλια:

Πολλοί από τους Lagrange multipliers $\alpha_i \geq 0$, είναι μηδέν.

- **w** είναι γραμμικός συνδυασμός λίγων δεδομένων

$$\mathbf{w} = \sum_{i=1}^{N_S} \alpha_i y_i \mathbf{x_i}$$

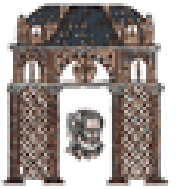
- με $N_s \leq N_0$, να αντιστοιχεί στους θετικούς Lagrange multipliers. Τα διανύσματα αυτά ονομάζονται support vectors

- Από το περιορισμό (4) έχουμε.,

$$\alpha_i [y_i (\underline{w}^T \underline{x}_i + w_0) - 1] = 0, \quad i = 1,2,..., N$$

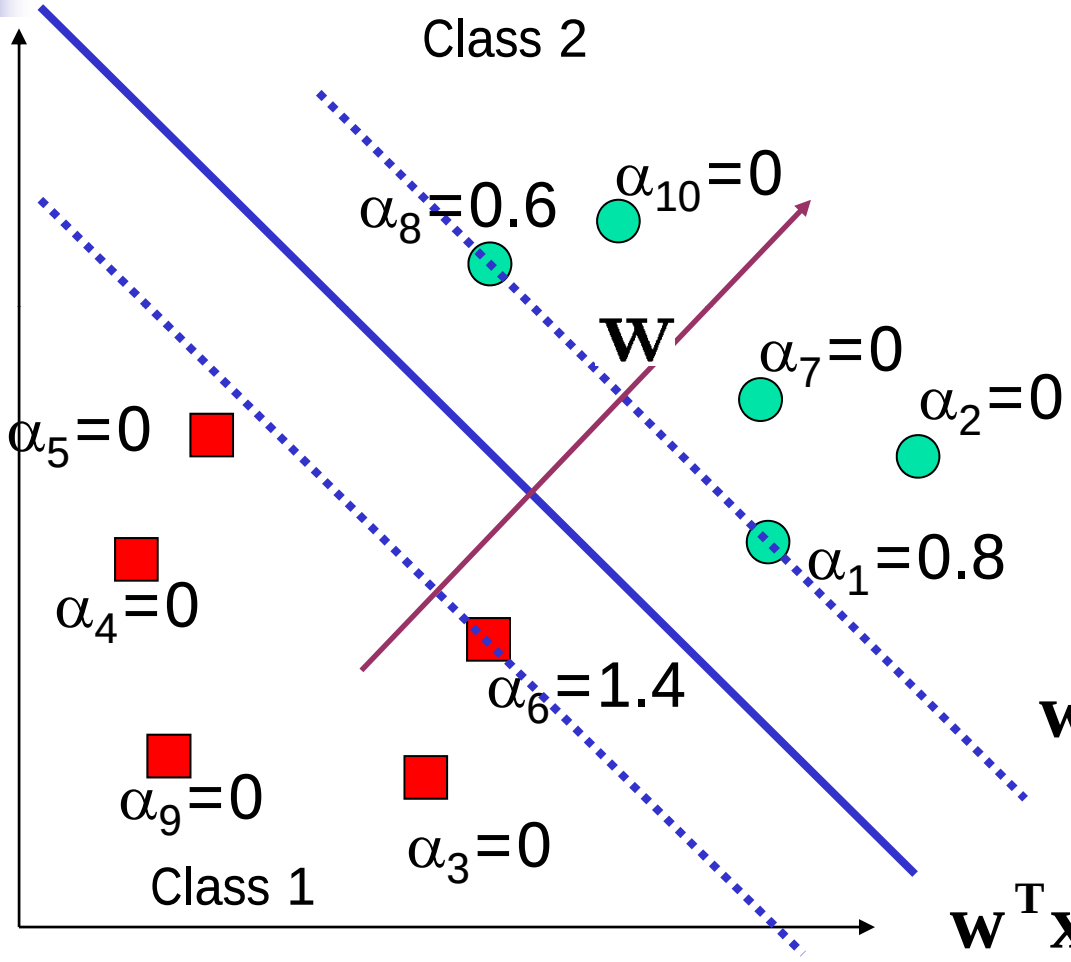
- Άρα για τα διανύσματα αυτά (SV) που συνεισφέρουν στο **w**, έχουμε

$$\mathbf{w}^T \mathbf{x}_i + w_0 = \pm 1 \quad i = 1,2,..., N_s \quad (4)$$



- Τα διανύσματα αυτά είναι γνωστά σαν ΔΙΑΝΥΣΜΑΤΑ ΣΤΗΡΙΞΗΣ (SUPPORT VECTORS) και είναι τα πλησιέστερα διανύσματα (δεδομένα), από κάθε κατηγορία στον ταξινομητή.
- Αφού υπολογίσουμε το $\boldsymbol{w}$, υπολογίζουμε μετά το $\boldsymbol{w_0}$ από την (4).
- Τα Διανύσματα Στήριξης (δεδομένα), τόσο στην εκπαίδευση όσο και στον έλεγχο, εισέρχονται μέσα από εσωτερικά γινόμενα $<x^T,y>$. Αυτό είναι σημαντικό στην γενίκευση των SVM
- Το υπερεπίπεδο του ταξινομητή που βρίσκουμε από μία support vector machine είναι ΜΟΝΑΔΙΚΟ.
- Παρ' όλο ότι η λύση είναι μοναδική, οι Lagrange multipliers ΔΕΝ ΕΙΝΑΙ.

Μία ΓΕΩΜΕΤΡΙΚΗ ΕΡΜΗΝΕΙΑ

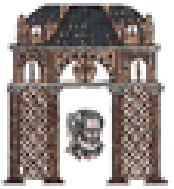


Συνεπώς, εάν αλλάξουμε τα σημεία στον εξωτερικό χώρο (a_i=0) τα διανύσματα στήριξης παραμένουν αναλλοίωτα.

Ορισμός του Δυαδικού προβλήματος

- Η μορφή των SVM είναι ένα πρόβλημα γραμμικού προγραμματισμού σε κυρτές επιφάνειες, με
  - Κυρτή επιφάνεια κόστους
  - Κυρτή περιοχή επιτρεπτών λύσεων
- Συνεπώς η λύση μπορεί να βρεθεί με την επίλυση του δυαδικού προβλήματος, δηλαδή,
  - Μεγιστοποίησε $L(\mathbf{w}, w_0, \boldsymbol{\alpha})$ ως προς $\boldsymbol{\alpha}$
  - Με τους περιορισμούς

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



- Συνδυάζοντας τα προηγούμενα έχουμε

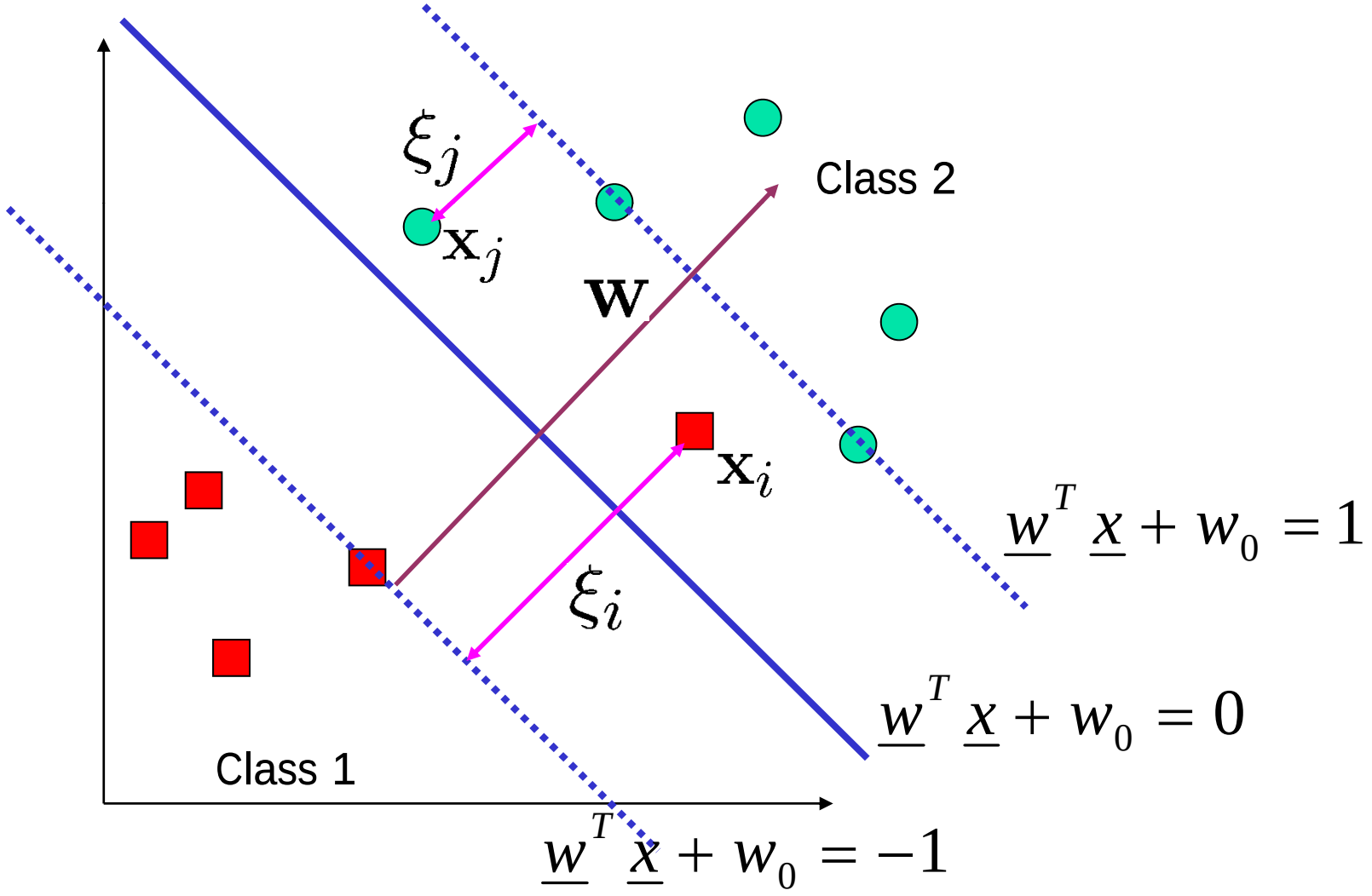
  - maximize $\underline{\alpha}$ $\quad (\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \underline{x}_i^T \underline{x}_j)$

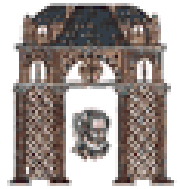
  - subject to $\quad \sum_{i=1}^{N} \alpha_i y_i = 0$

    $\underline{\alpha} \geq \underline{0}$

Και εάν δεν είναι γραμμικά διαχωρίσιμες κατηγορίες?

- Επιτρέπουμε ένα λάθος ξ_i στην ταξινόμηση





Μη γραμμικά διαχωρίσιμες κατηγορίες

- Στην περίπτωση αυτή δεν υπάρχει υπερεπίπεδο w έτσι ώστε

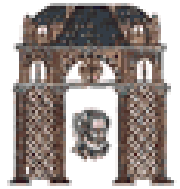
$$\underline{w}^T \underline{x} + w_0 (>,<) 1, \quad \forall \underline{x}$$

- Σημειώνεται ότι το περιθώριο έχει ορισθεί σαν την απόσταση ανάμεσα στα 2 υπερεπίπεδα

$$\underline{w}^T \underline{x} + w_0 = 1$$

$$\underline{w}^T \underline{x} + w_0 = -1$$

Σχόλια: Παρ' όλο ότι η λύση, $\underline{\boldsymbol{w}}$, είναι ΜΟΝΑΔΙΚΗ, οι πολλαπλασιαστές Lagrange ΔΕΝ ΕΙΝΑΙ



Μη γραμμικά διαχωρίσιμες κατηγορίες

- Τα διανύσματα εκπαίδευση ανήκουν σε μία από τις τρείς πιθανές κατηγορίες

  - Α. Διανύσματα έξω από την λωρίδα και τα οποία ταξινομούνται σωστά, δηλαδή

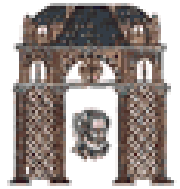
$$y_i(\underline{w}^T \underline{x} + w_0) > 1$$

  - Β. Διανύσματα μέσα στην λωρίδα τα οποία ταξινομούνται πάλι σωστά, δηλαδή

$$0 \le y_i(\underline{w}^T \underline{x} + w_0) < 1$$

  - C. Διανύσματα λάθος ταξινομημένα, δηλαδή

$$y_i(\underline{w}^T \underline{x} + w_0) < 0$$



Μη γραμμικά διαχωρίσιμες κατηγορίες

- Οι τρείς αυτές κατηγορίες μπορούμε να τις γράψουμε

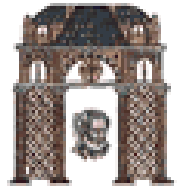
$$y_i(\underline{w}^T \underline{x} + w_0) \geq 1 - \xi_i$$

A. $\rightarrow \xi_i = 0$

B. $\rightarrow 0 < \xi_i \leq 1$

C. $\rightarrow 1 < \xi_i$

Τα ξ_i είναι γνωστά σαν slack μεταβλητές



Μη γραμμικά διαχωρίσιμες κατηγορίες

Τώρα πρέπει να βελτιστοποιήσουμε δύο (2) μεγέθη

- Μεγιστοποιήσουμε το περιθώριο
- Και να ελαχιστοποιήσουμε των αριθμό των προτύπων για τα οποία $\xi_i > 0$

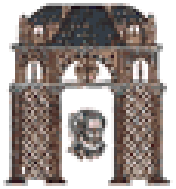
  Δηλαδή

$$J(\underline{w}, w_0, \underline{\xi}) = \frac{1}{2}\|\underline{w}\|^2 + C\sum_{i=1}^{N} I(\xi_i)$$

  όπου το C είναι μία σταθερά

$$I(\xi_i) = \begin{cases} 1 & \xi_i > 0 \\ 0 & \xi_i = 0 \end{cases}$$

- I(.) δεν είναι παραγωγίσιμο. Στην πράξη, όπως κάναμε και για τα perceptron, χρησιμοποιούμε την προσέγγιση
- $$J(\underline{w}, w_0, \underline{\xi}) = \frac{1}{2}\|\underline{w}\|^2 + C\sum_{i=1}^{N} \xi_i$$
- Ακολουθώντας την ίδια διαδικασία με την προηγούμενη περίπτωση, παίρνουμε



ΚΚΤ συνθήκες

$$(1)\ \ \underline{w} = \sum_{i=1}^{N} \lambda_i y_i \underline{x}_i$$

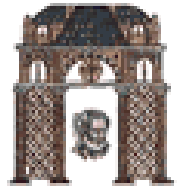
$$(2)\ \ \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$(3)\ \ C - \mu_i - \lambda_i = 0, i = 1,2,..., N$$

$$(4)\ \ \lambda_i [y_i (\underline{w}^T \underline{x}_i + w_0) - 1 + \xi_i] = 0, \quad i = 1,2,..., N$$

$$(5)\ \ \mu_i \xi_i = 0, \quad i = 1,2,..., N$$

$$(6)\ \ \mu_i, \lambda_i \geq 0, \quad i = 1,2,..., N$$



Μη γραμμικά διαχωρίσιμες κατηγορίες

- Το αντίστοιχο δυαδικό πρόβλημα είναι

  Μεγιστοποίησε το $\underline{\alpha}(\sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \underline{x}_i^T \underline{x}_j)$

  με τον περιορισμό

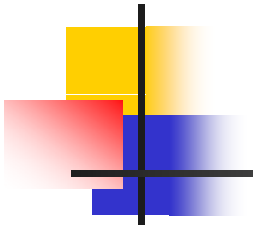
  $$0 \le \alpha_i \le C,\ i = 1,2,\ldots,N$$

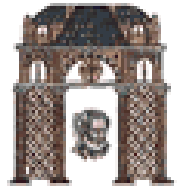
  $$\sum_{i=1}^{N}\alpha_i y_i = 0$$

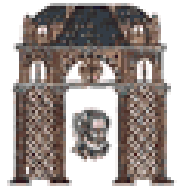
- <u>Σχόλιο</u>:

  Η μόνη διαφορά με την γραμμικά διαχωρίσιμη περίπτωση είναι η ύπαρξη της σταθεράς C στους περιορισμούς

Μη γραμμικοί ταξινομητές με μηχανές διανυσματικής στήριξης

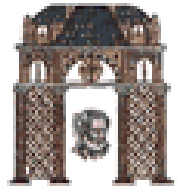






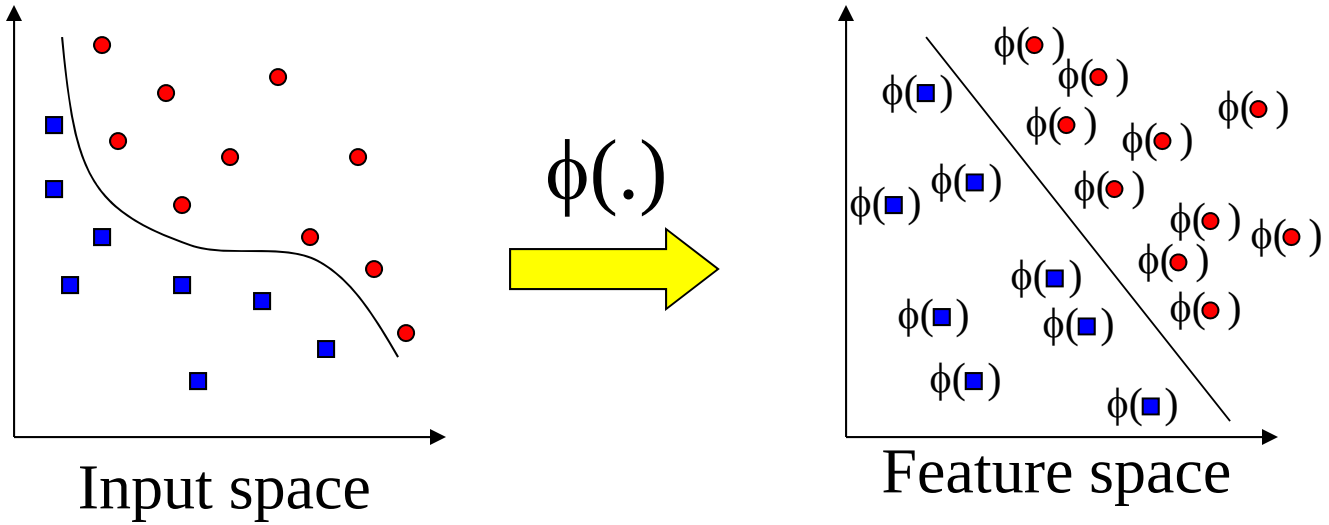
Extension to Non-linear Decision Boundary

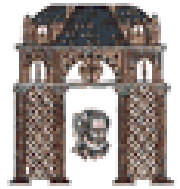
- Key idea: transform $\mathbf{x}_i$ to a higher dimensional space to "make life easier"
  - Input space: the space $\mathbf{x}_i$ are in
  - Feature space: the space of $\phi(\mathbf{x}_i)$ after transformation
- Why transform?
  - Linear operation in the feature space is equivalent to non-linear operation in input space
  - The classification task can be "easier" with a proper transformation. Example: XOR



Extension to Non-linear Decision Boundary

- Possible problem of the transformation
  - High computation burden and hard to get a good estimate
- SVM solves these two issues simultaneously
  - Kernel tricks for efficient computation
  - Minimize $||\mathbf{w}||^2$ can lead to a “good” classifier





Example Transformation

- Define the kernel function $K(\mathbf{x},\mathbf{y})$ as

$$K(\mathbf{x},\mathbf{y}) = (1 + x_1y_1 + x_2y_2)^2$$

- Consider the following transformation

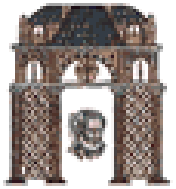
$$\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) = (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1y_2)$$

$$\langle \phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}), \phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) \rangle = (1 + x_1y_1 + x_2y_2)^2$$

$$= K(\mathbf{x},\mathbf{y})$$

- The inner product can be computed by going through the map $\phi(.)$

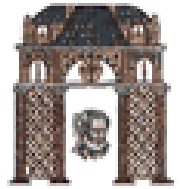


Kernel Trick

- The relationship between the kernel function *K* and the mapping $\phi(.)$ is

$$K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$$

  - This is known as the kernel trick
- In practice, we specify *K*, thereby specifying $\phi(.)$ indirectly, instead of choosing $\phi(.)$
- Intuitively, *K* (**x**,**y**) represents our desired notion of similarity between data **x** and **y** and this is from our prior knowledge
- *K* (**x**,**y**) needs to satisfy a technical condition (Mercer condition) in order for $\phi(.)$ to exist



Examples of Kernel Functions

- Polynomial kernel with degree *d*

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^d$$

- Radial basis function kernel with width σ

$$K(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/(2\sigma^2))$$

  - Closely related to radial basis function neural networks

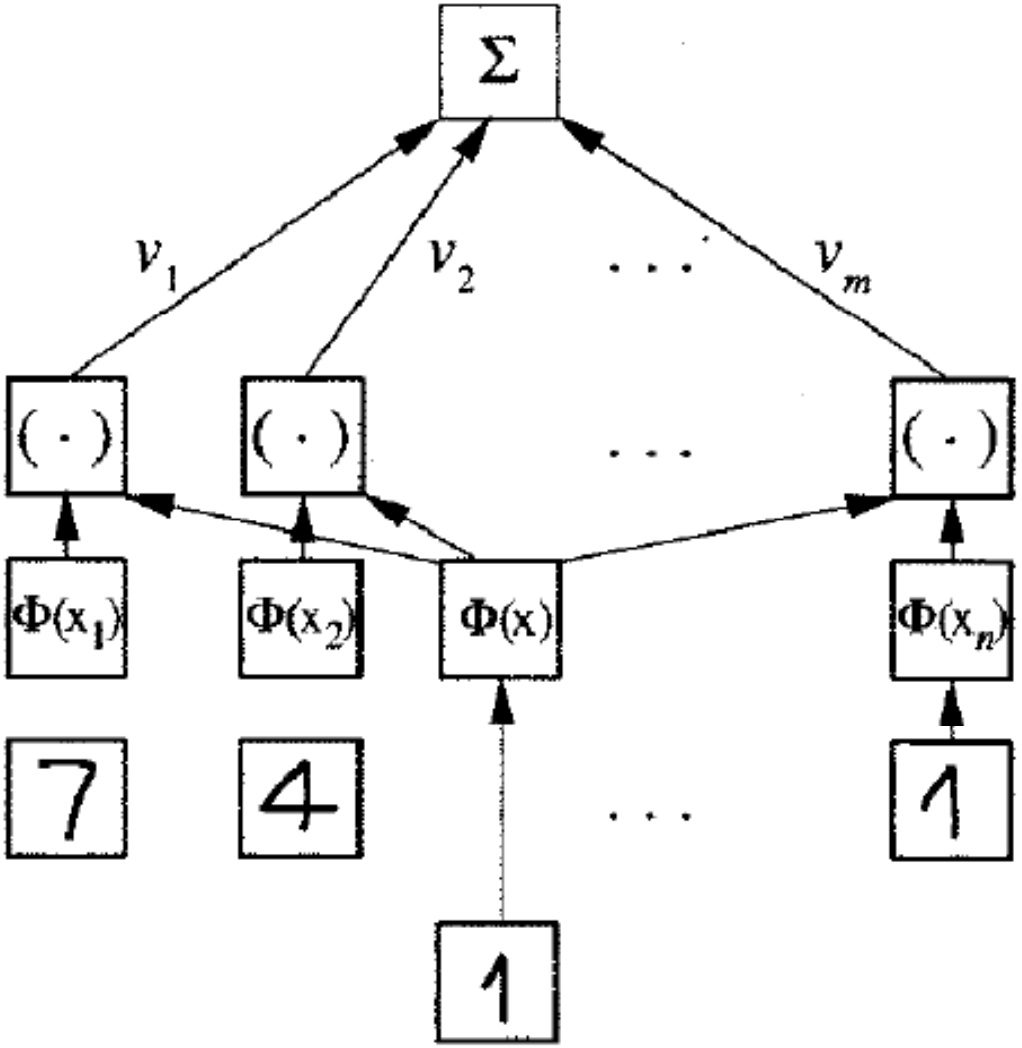
- Sigmoid with parameter κ and θ

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa\mathbf{x}^T\mathbf{y} + \theta)$$

  - It does not satisfy the Mercer condition on all κ and θ

- Research on different kernel functions in different applications is very active

Example of SVM Applications: Handwriting Recognition



output $\Sigma\, v_i\, k\,(x,x_i) + b$

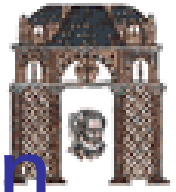
weights

dot product $(\Phi(x)\cdot\Phi(x_i)) = k(x,x_i)$

mapped vectors $\Phi(x_i)$, $\Phi(x)$

support vectors $x_1 \ldots x_n$

test vector x



Modification Due to Kernel Function

- Change all inner products to kernel functions
- For training,

Original

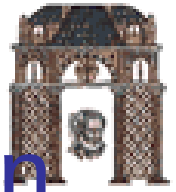
$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

With kernel function

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

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$



Modification Due to Kernel Function

- For testing, the new data **z** is classified as class 1 if $f \geq 0$, and as class 2 if $f < 0$

Original

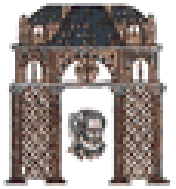
$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

$$f = \mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}^T \mathbf{z} + b$$

With kernel function

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \phi(\mathbf{x}_{t_j})$$

$$f = \langle \mathbf{w}, \phi(\mathbf{z}) \rangle + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} K(\mathbf{x}_{t_j}, \mathbf{z}) + b$$

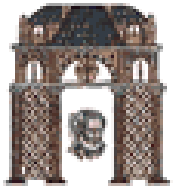


Example

- Suppose we have 5 1D data points
  - x_1=1, x_2=2, x_3=4, x_4=5, x_5=6, with 1, 2, 6 as class 1 and 4, 5 as class 2 $\Rightarrow$ y_1=1, y_2=1, y_3=-1, y_4=-1, y_5=1
- We use the polynomial kernel of degree 2
  - $K(x,y) = (xy+1)^2$
  - C is set to 100
- We first find α_i (i=1, …, 5) by

$$\text{max.} \sum_{i=1}^{5} \alpha_i - \frac{1}{2} \sum_{i=1}^{5} \sum_{j=1}^{5} \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$$

$$\text{subject to } 100 \geq \alpha_i \geq 0, \sum_{i=1}^{5} \alpha_i y_i = 0$$



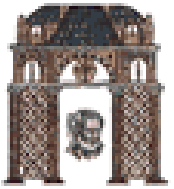
Example

- By using a QP solver, we get
  - α_1=0, α_2=2.5, α_3=0, α_4=7.333, α_5=4.833
  - Note that the constraints are indeed satisfied
  - The support vectors are {x_2=2, x_4=5, x_5=6}
- The discriminant function is

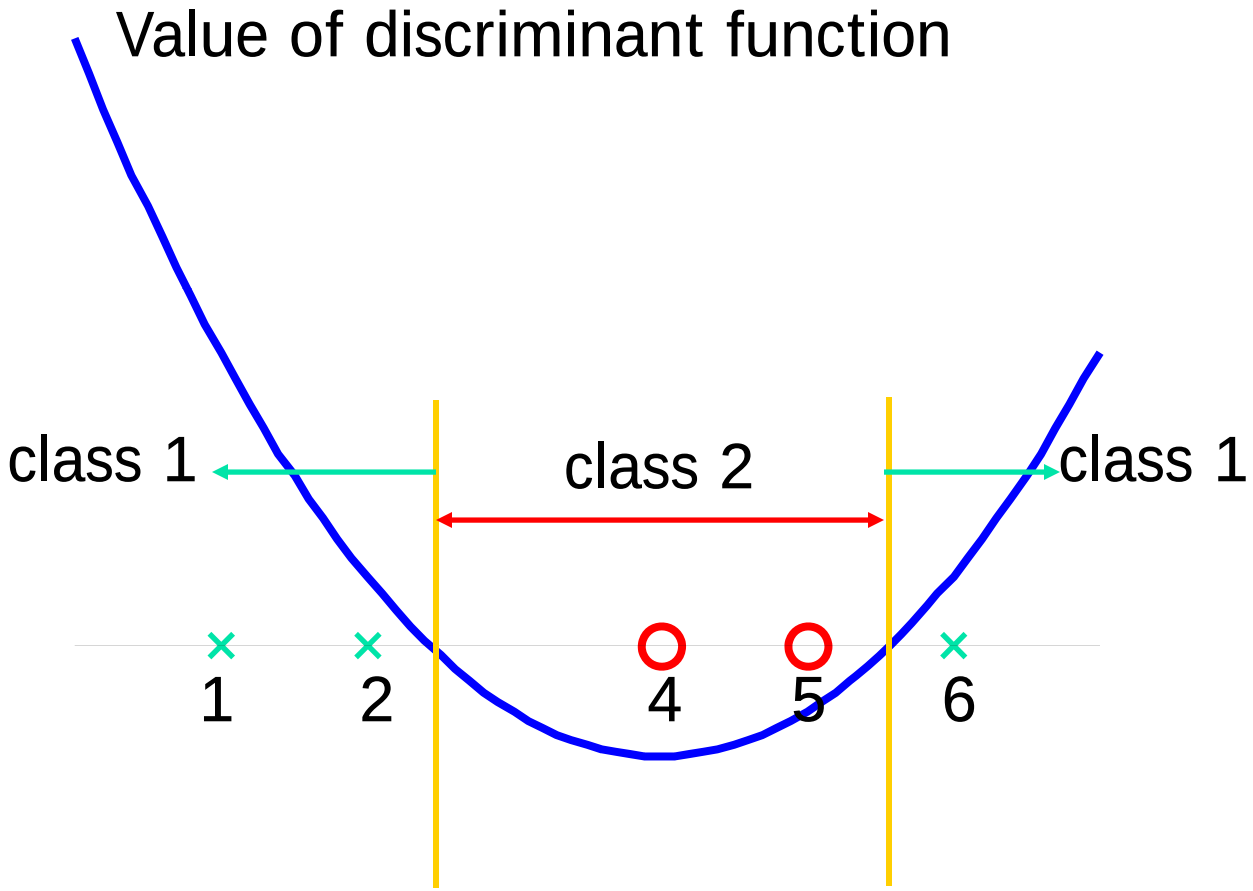
$$f(y) = 2.5(1)(2y+1)^2 + 7.333(-1)(5y+1)^2 + 4.833(1)(6y+1)^2 + b$$
$$= 0.6667x^2 - 5.333x + b$$

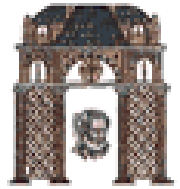
- *b* is recovered by solving f(2)=1 or by f(5)=-1 or by f(6)=1, as x_2, x_4, x_5 lie on $y_i(\mathbf{w}^T\phi(z)+b)=1$ and all give b=9

$$\rightarrow \quad f(y) = 0.6667x^2 - 5.333x + 9$$



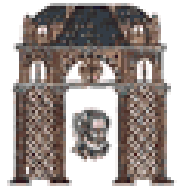
Example


Value of discriminant function
class 1
class 2
class 1
1
2
4
5
6



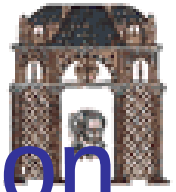
Multi-class Classification

- SVM is basically a two-class classifier
- One can change the QP formulation to allow multi-class classification
- More commonly, the data set is divided into two parts “intelligently” in different ways and a separate SVM is trained for each way of division
- Multi-class classification is done by combining the output of all the SVM classifiers
  - Majority rule
  - Error correcting code
  - Directed acyclic graph



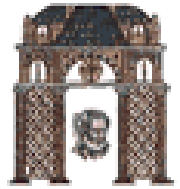
Software

- A list of SVM implementation can be found at http://www.kernel-machines.org/software.html
- Some implementation (such as LIBSVM) can handle multi-class classification
- SVMLight is among one of the earliest implementation of SVM
- Several Matlab toolboxes for SVM are also available



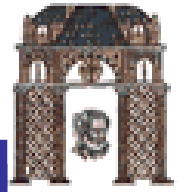
Summary: Steps for Classification

- Prepare the pattern matrix
- Select the kernel function to use
- Select the parameter of the kernel function and the value of C
  - You can use the values suggested by the SVM software, or you can set apart a validation set to determine the values of the parameter
- Execute the training algorithm and obtain the α_i
- Unseen data can be classified using the α_i and the support vectors



Demonstration

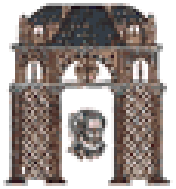
- Iris data set
  - Class 1 and class 3 are “merged” in this demo



Strengths and Weaknesses of SVM

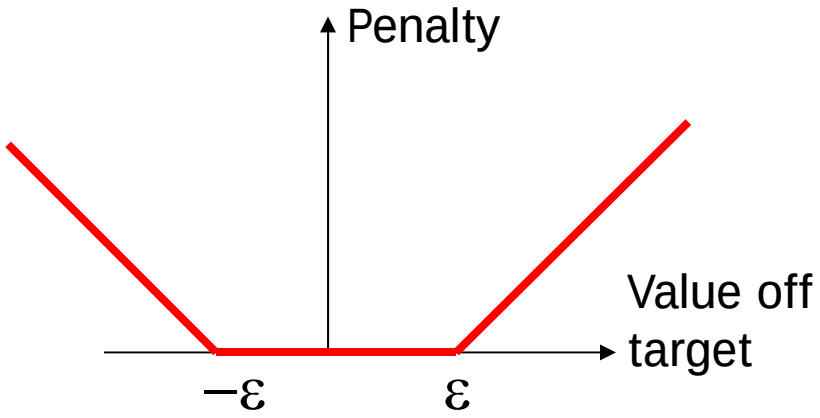
- Strengths
  - Training is relatively easy
    - No local optimal, unlike in neural networks
  - It scales relatively well to high dimensional data
  - Tradeoff between classifier complexity and error can be controlled explicitly
  - Non-traditional data like strings and trees can be used as input to SVM, instead of feature vectors
- Weaknesses
  - Need a “good” kernel function

Epsilon Support Vector Regression (ε-SVR)

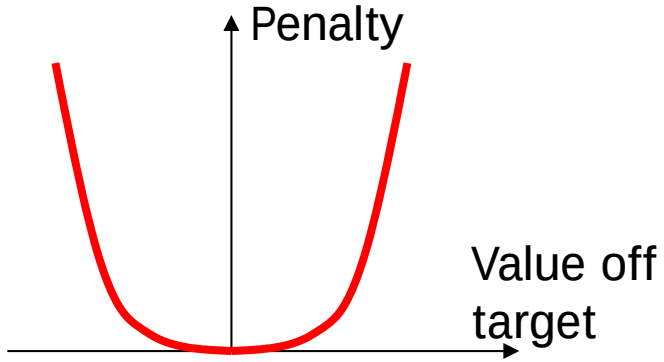


- Linear regression in feature space
- Unlike in least square regression, the error function is ε-insensitive loss function
  - Intuitively, mistake less than ε is ignored
  - This leads to sparsity similar to SVM

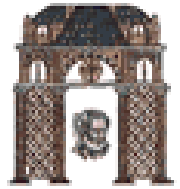
ε-insensitive loss function



Square loss function



Epsilon Support Vector Regression (ε-SVR)

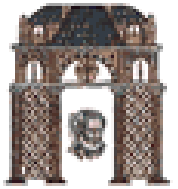


- Given: a data set $\{x_1, ..., x_n\}$ with target values $\{u_1, ..., u_n\}$, we want to do ε-SVR
- The optimization problem is

$$\text{Min } \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$

$$\text{subject to } \begin{cases} u_i - \mathbf{w}^T\mathbf{x}_i - b \leq \epsilon + \xi_i \\ \mathbf{w}^T\mathbf{x}_i + b - u_i \leq \epsilon + \xi_i^* \\ \xi_i \geq 0, \xi_i^* \geq 0 \end{cases}$$

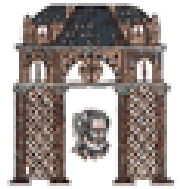
- Similar to SVM, this can be solved as a quadratic programming problem



Epsilon Support Vector Regression (ε-SVR)

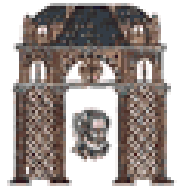
- *C* is a parameter to control the amount of influence of the error
- The ½||w||^2 term serves as controlling the complexity of the regression function
  - This is similar to ridge regression
- After training (solving the QP), we get values of α_i and α_i^*, which are both zero if $\mathbf{x}_i$ does not contribute to the error function
- For a new data **z**,

$$f(\mathbf{z}) = \sum_{j=1}^{s} (\alpha_{t_j} - \alpha_{t_j}^*) K(\mathbf{x}_{t_j}, \mathbf{z}) + b$$



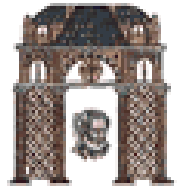
Other Types of Kernel Methods

- A lesson learnt in SVM: a linear algorithm in the feature space is equivalent to a non-linear algorithm in the input space
- Classic linear algorithms can be generalized to its non-linear version by going to the feature space
  - Kernel principal component analysis, kernel independent component analysis, kernel canonical correlation analysis, kernel k-means, 1-class SVM are some examples



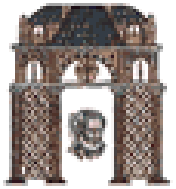
SVM vs. Neural Networks

- SVM
  - Relatively new concept
  - Nice Generalization properties
  - Hard to learn – learned in batch modes using QP techniques
  - Using kernels can learn very complex functions
- Neural Networks
  - Generalizes well but doesn't have mathematical foundation
  - Can easily be learnt in incremental fashion
  - To learn complex function – use complex multi layer structure.



Conclusion

- SVM is a useful alternative to neural networks for classification
- Two key concepts of SVM: maximize the margin and the kernel trick
- Many active research is taking place on areas related to SVM
- Many SVM implementations are available on the web for you to try on your data set!



Resources

- http://www.kernel-machines.org/
- http://www.support-vector.net/
- http://www.support-vector.net/icml-tutorial.pdf
- http://www.kernel-machines.org/papers/tutorial-nips.ps.gz
- http://www.clopinet.com/isabelle/Projects/SVM/applist.html