Μηχανές Διανυσμάτων Υποστήριξης Support Vector Machines

Κώστας Διαμαντάρας
Τμήμα Πληροφορικής
ΤΕΙ Θεσσαλονίκης

Γραμμικός διαχωρισμός 2 κλάσεων

- Ξαναμελετάμε το πρόβλημα του γραμμικού διαχωρισμού 2 κλάσεων C_0, C_1:
- Δεδομένα: διανύσματα εισόδου (πρότυπα) $\mathbf{x}_k$ και οι αντίστοιχοι στόχοι τους $d_k = -1$ ή 1.

$$\{x_1,d_1\}, \{x_2,d_2\}, \ldots, \{x_N,d_N\}.$$

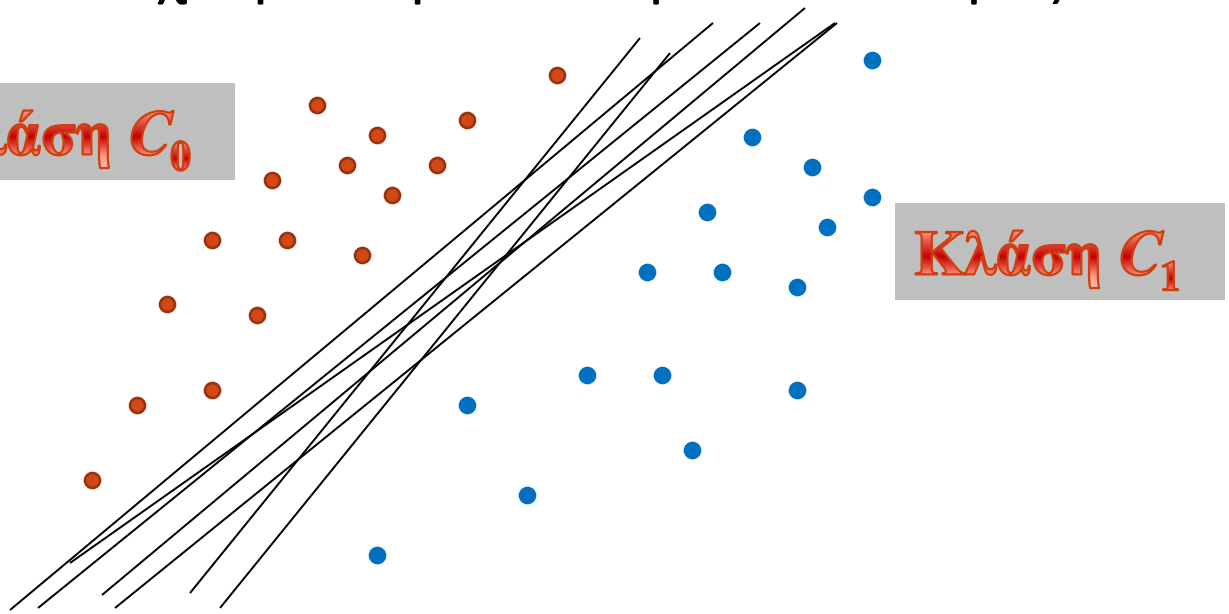
- Ζητούμενο: να βρεθεί διάνυσμα βαρών $\mathbf{w}$ και πόλωση w_0 ώστε

$$\mathbf{w}^T\mathbf{x}_k + w_0 < 0, \quad \text{αν } \mathbf{x}_k \in C_0$$

$$\mathbf{w}^T\mathbf{x}_k + w_0 > 0, \quad \text{αν } \mathbf{x}_k \in C_1$$

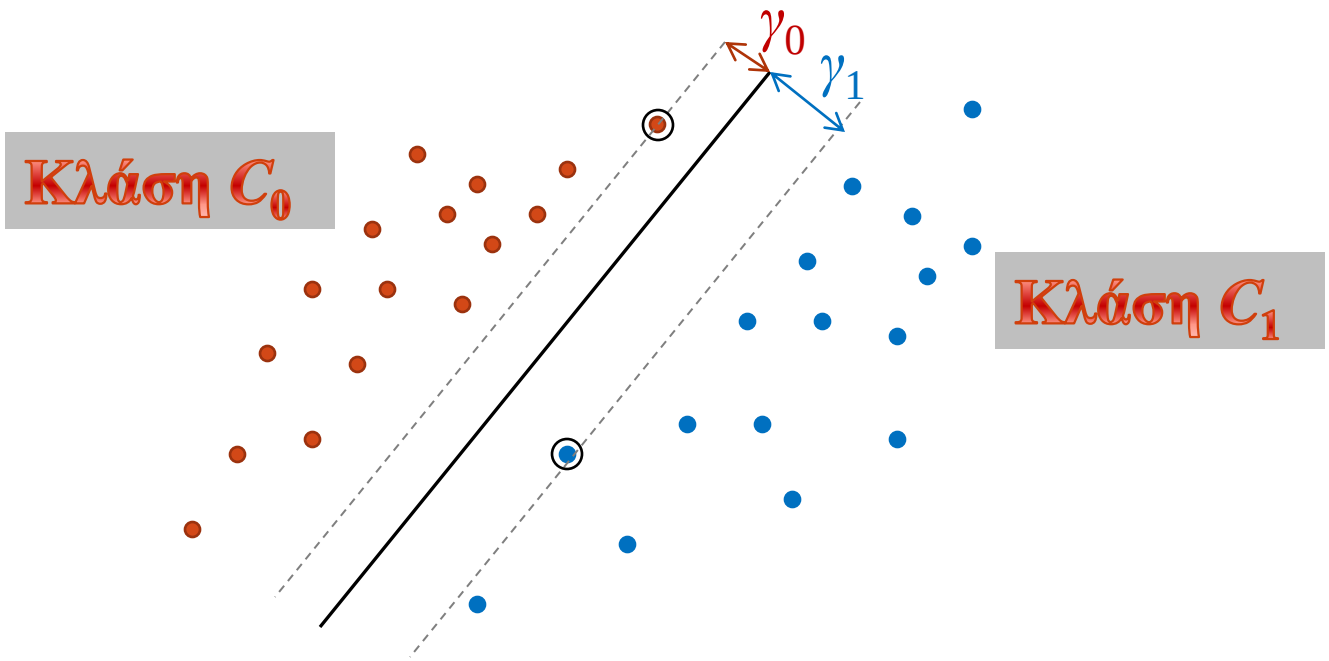
Γραμμικά διαχωρίσιμο πρόβλημα

- Ουσιαστικά αναζητείται μια ευθεία που να διαχωρίζει τις δύο κλάσεις
- Αν το πρόβλημα έχει λύση (γραμμικά διαχωρίσιμο) τότε δεν έχει μόνο μια λύση αλλά άπειρες



Περιθώριο ταξινόμησης

- Ορίζουμε γ_0 (γ_1) την απόσταση της διαχωριστικής ευθείας από το κοντινότερο πρότυπο της κλάσης C_0 (C_1)



Περιθώριο ταξινόμησης (2)

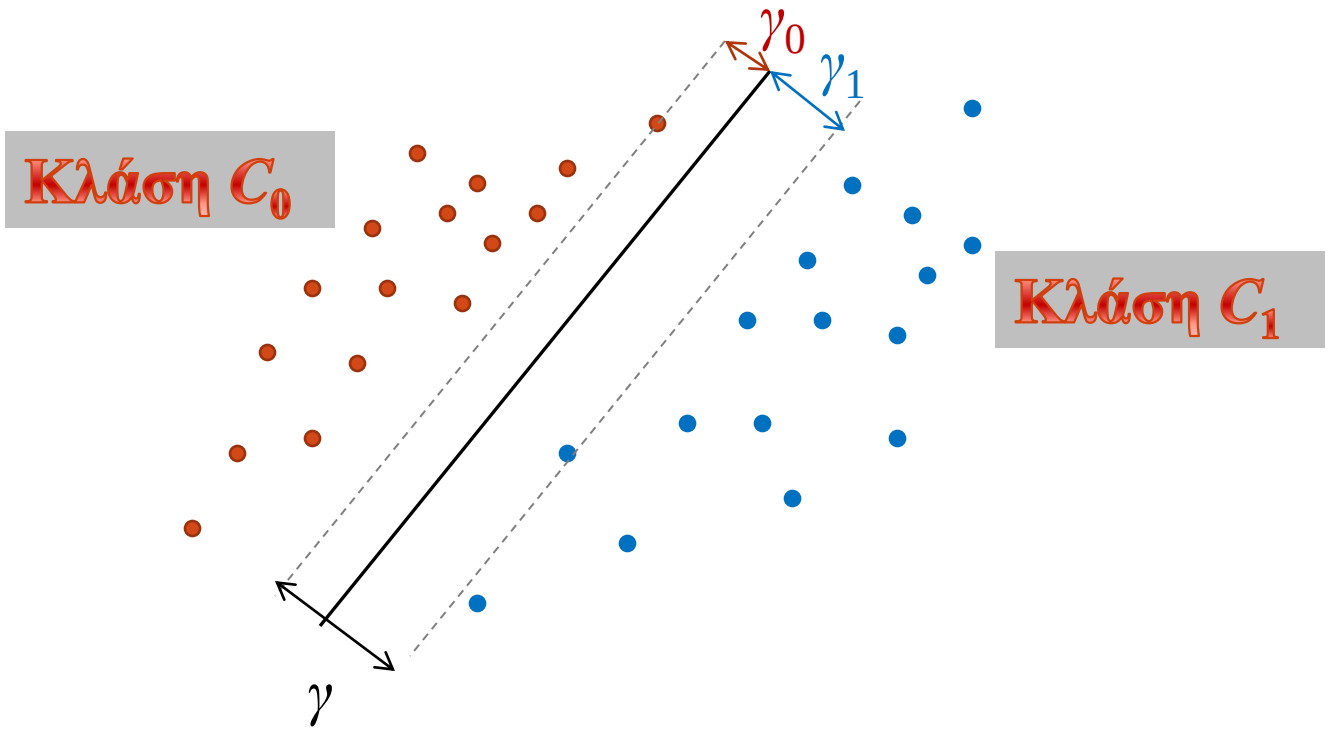
$$\gamma_0 = \min_{\mathbf{x} \in C_0} \frac{\left| \mathbf{w}^T \mathbf{x} + w_0 \right|}{\left\| \mathbf{w} \right\|}$$

$$\gamma_1 = \min_{\mathbf{x} \in C_1} \frac{\left| \mathbf{w}^T \mathbf{x} + w_0 \right|}{\left\| \mathbf{w} \right\|}$$

- Περιθώριο $\gamma = \gamma_0 + \gamma_1$

Περιθώριο ταξινόμησης (3)

- Περιθώριο γ



Κανονικό διαχωριστικό υπερεπίπεδο

- Επέλεξε w_0 ώστε $\gamma_0 = \gamma_1$
- Κλιμάκωσε $\mathbf{w}$ ώστε

$$\min_{\mathbf{x}\in C_0}\left|\mathbf{w}^T\mathbf{x}+w_0\right| = \min_{\mathbf{x}\in C_1}\left|\mathbf{w}^T\mathbf{x}+w_0\right| = 1$$

- Άρα

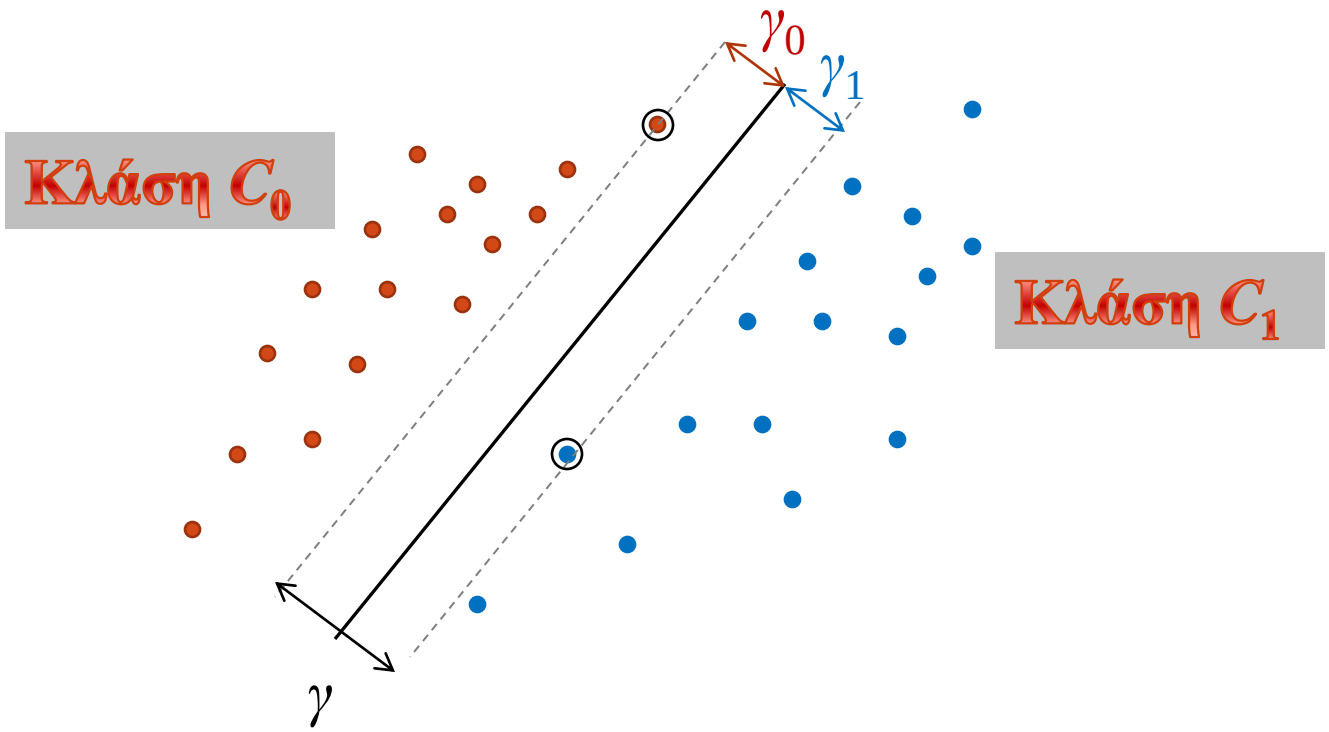
$$\mathbf{w}^T\mathbf{x}_k + w_0 \le -1, \ \text{αν} \ \mathbf{x}_k \in C_0$$

$$\mathbf{w}^T\mathbf{x}_k + w_0 \ge 1, \ \text{αν} \ \mathbf{x}_k \in C_1$$

- Αφού $d_k=-1$, αν $\mathbf{x}_k\in C_0$, $d_k=1$, αν $\mathbf{x}_k\in C_1$, τότε απλά

$$d_k(\mathbf{w}^T\mathbf{x}_k + w_0) \ge 1$$

Διανύσματα υποστήριξης (support vectors)



- Περιθώριο

$$\gamma = \frac{2}{\|\mathbf{w}\|}$$

Δύο κατηγορίες προτύπων

- Ορισμός

  Διανύσματα υποστήριξης (support vectors):

  Τα πρότυπα $\mathbf{x}_k$ για τα οποία

$$d_k(\mathbf{w}^T\mathbf{x}_k + w_0) = 1$$

- Όλα τα υπόλοιπα πρότυπα:

  Τα πρότυπα $\mathbf{x}_k$ για τα οποία

$$d_k(\mathbf{w}^T\mathbf{x}_k + w_0) > 1$$

Βέλτιστο Διαχωριστικό Υπερεπίπεδο

- Πρόβλημα βέλτιστου διαχωριστικού υπερεπιπέδου:

  Βρες το υπερεπίπεδο που μεγιστοποιεί το περιθώριο. Ισοδύναμα, ελαχιστοποίησε τη συνάρτηση κόστους:

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

  Φυσικά πρέπει να ικανοποιούνται όλες οι ανισότητες

  για k=1, …, N. $d_k(\mathbf{w}^T\mathbf{x}_k + w_0) \geq 1$

Τετραγωνικός Προγραμματισμός

- Πρόβλημα τετραγωνικού προγραμματισμού:

Βρες το ελάχιστο της συνάρτησης

$$J_{\tau\pi}(\mathbf{v}) = \frac{1}{2}\mathbf{v}^T\mathbf{H}\mathbf{v} + \mathbf{f}^T\mathbf{v}$$

υπό τους περιορισμούς $\mathbf{a}_k^T\mathbf{v} \le b_k, \quad k = 1, \cdots, N$

$$\text{συνοπτικά}: \mathbf{A}\mathbf{v} \le \mathbf{b}$$

- Πρόβλημα γνωστό εδώ και δεκαετίες
- MATLAB function: `quadprog(H,f,A,b)`

Βέλτιστο Διαχωριστικό Υπερεπίπεδο = Τετραγωνικός Προγραμματισμός

- Τα δύο προβλήματα είναι ίδια. Αρκεί να βάλουμε:

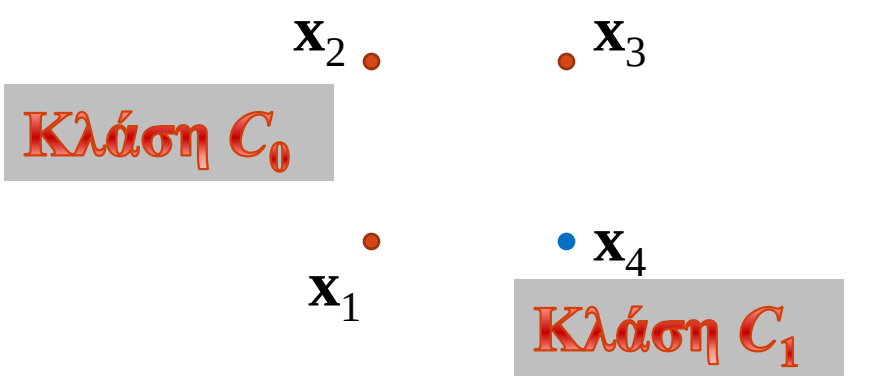
$$\mathbf{v} = \begin{bmatrix} \mathbf{w} \\ w_0 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} \mathbf{I} & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{f} = 0,$$

$$\mathbf{A} = -\begin{bmatrix} d_1 \mathbf{x}_1^T & d_1 \\ \vdots & \vdots \\ d_1 \mathbf{x}_N^T & d_1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -1 \\ \vdots \\ -1 \end{bmatrix}$$

- Η λύση μπορεί να δοθεί άμεσα από το MATLAB

Παράδειγμα

- Θέλουμε να διαχωρίσουμε τις παρακάτω κλάσεις



$$\mathbf{x}_1 = [0,0]^T, \quad d_1 = -1$$
$$\mathbf{x}_2 = [0,1]^T, \quad d_2 = -1$$
$$\mathbf{x}_3 = [1,1]^T, \quad d_3 = -1$$
$$\mathbf{x}_4 = [1,0]^T, \quad d_4 = 1$$

- Θέτουμε:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{A} = -\begin{bmatrix} 0 & 0 & -1 \\ 0 & -1 & -1 \\ -1 & -1 & -1 \\ 1 & 0 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$$

Εναλλακτική λύση: δυϊκό πρόβλημα

- Αντί να λύσουμε το πρόβλημα ως έχει, υποθέτουμε ότι η λύση είναι ένας γραμμικός συνδυασμός των προτύπων

$$\mathbf{w} = \sum_{k=1}^{N} \lambda_k d_k \mathbf{x}_k$$

οπότε αποδεικνύεται ότι

$$\lambda_k \geq 0, \quad k = 1, \cdots, N, \quad \text{και} \quad \sum_{k=1}^{N} d_k \lambda_k = 0$$

και το πρόβλημά μας είναι μαθηματικά ισοδύναμο με την λύση του λεγόμενου δυϊκού προβλήματος:

Το δυϊκό πρόβλημα

- Δυϊκό Πρόβλημα τετραγωνικού προγραμματισμού:

Βρες το ελάχιστο της συνάρτησης

$$L(\boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{\lambda}^T \mathbf{Q}\boldsymbol{\lambda} + \mathbf{g}^T\boldsymbol{\lambda}$$

υπό τους περιορισμούς $\lambda_k \geq 0, \quad k = 1, \cdots, N$

$$\sum_{k=1}^{N} d_k \lambda_k = 0 \quad \text{ισοδ.} \quad \mathbf{d}^T\boldsymbol{\lambda} = 0$$

όπου

$$\mathbf{Q} : q_{ij} = d_i d_j \mathbf{x}_i^T \mathbf{x}_j \qquad \mathbf{g} : g_i = -1$$

Δυϊκό πρόβλημα - Ιδιότητες

- Πάλι έχουμε ένα πρόβλημα τετραγωνικού προγραμματισμού αλλά με άλλους πίνακες.
- Αποδεικνύεται:

  είτε $0 < \lambda_k \quad \Leftrightarrow \quad d_k(\mathbf{w}^T\mathbf{x}_k + w_0) = 1$ οπότε το $\mathbf{x}_k$ είναι διάνυσμα υποστήριξης

  είτε $0 = \lambda_k \quad \Leftrightarrow \quad d_k(\mathbf{w}^T\mathbf{x}_k + w_0) > 1$
- Συνεπώς το βέλτιστο **w** είναι γραμμικός συνδυασμός των δ.υ. και μόνο
- Επίσης για οποιοδήποτε δ.υ. η βέλτιστη πόλωση είναι

$$w_0 = \frac{1}{d_k} - \mathbf{w}^T\mathbf{x}_k$$

Μη γραμμικά διαχωρίσιμα προβλήματα

- Έστω ότι το πρόβλημα δεν διαχωρίζεται με ένα γραμμικό υπερεπίπεδο
- Εισάγουμε για κάθε πρότυπο μια μεταβλητή χαλαρότητας ξ_k

$$d_k(\mathbf{w}^T\mathbf{x}_k + w_0) \geq 1 - \xi_k$$

$$\xi_k \geq 0$$

- Αν $\xi_k > 1$ τότε το πρότυπο έχει ταξινομηθεί λάθος.

$$\sum_{k=1}^{N} \xi_k > \text{πλήθος λάθος ταξινομημένων προτύπων}$$

Νέα συνάρτηση κόστους

- Στην αρχική συνάρτηση κόστους προσθέτουμε ένα κόστος ανάλογο του $\sum_{k=1}^{N} \xi_k$
- Νέο πρόβλημα τετραγωνικού προγραμματισμού: ελαχιστοποίησε τη συνάρτηση κόστους:

$$J'(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{k=1}^{N} \xi_k$$

ώστε να ικανοποιούνται οι ανισότητες

$$d_k(\mathbf{w}^T\mathbf{x}_k + w_0) \geq 1 - \xi_k$$

$$\xi_k \geq 0$$

Λύση νέου προβλήματος

- Νέο διάνυσμα αγνώστων $\mathbf{v} = [\mathbf{w}^T, w_0, \boldsymbol{\xi}^T]^T$
- Νέο σετ πινάκων

$$\mathbf{H} = \begin{bmatrix} \mathbf{I} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{f} = [0, 0, C\mathbf{1}^T], \quad \mathbf{A} = -\begin{bmatrix} d_1\mathbf{x}_1^T & d_1 & 1 & & 0 \\ \vdots & \vdots & & \ddots & \\ d_N\mathbf{x}_N^T & d_N & 0 & & 1 \\ 0 & 0 & 1 & & 0 \\ \vdots & \vdots & & \ddots & \\ 0 & 0 & 0 & & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -1 \\ \vdots \\ -1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

- Ίδια λύση MATLAB: `quadprog(H,f,A,b)`

Νέο δυϊκό πρόβλημα

- Ομοίως διατυπώνεται το νέο δυϊκό πρόβλημα

Βρες το ελάχιστο της συνάρτησης

$$L(\boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{\lambda}^T \mathbf{Q}\boldsymbol{\lambda} + \mathbf{g}^T\boldsymbol{\lambda}$$

υπό τους περιορισμούς $C \geq \lambda_k \geq 0, \quad k = 1, \cdots, N$

$$\sum_{k=1}^{N} d_k \lambda_k = 0 \quad \text{ισοδ.} \quad \mathbf{d}^T\boldsymbol{\lambda} = 0$$

όπου

$$\mathbf{Q}: q_{ij} = d_i d_j \mathbf{x}_i^T \mathbf{x}_j \qquad \mathbf{g}: g_i = -1$$

Νέο δυϊκό πρόβλημα - Λύση

- είτε $0 < \lambda_k < C \Leftrightarrow d_k(\mathbf{w}^T\mathbf{x}_k + w_0) = 1$
  συνεπώς το $\mathbf{x}_k$ είναι διάνυσμα υποστήριξης
- είτε $\lambda_k = 0 \Leftrightarrow d_k(\mathbf{w}^T\mathbf{x}_k + w_0) > 1$
  συνεπώς το $\mathbf{x}_k$ ταξινομείται σωστά αλλά δεν είναι διάνυσμα υποστήριξης
- είτε $\lambda_k = C \Leftrightarrow d_k(\mathbf{w}^T\mathbf{x}_k + w_0) < 1$
  συνεπώς το $\mathbf{x}_k$ ίσως ταξινομείται σωστά ίσως ταξινομείται λάθος αλλά δεν είναι διάνυσμα υποστήριξης

Νέο δυϊκό πρόβλημα – Λύση (συν.)

- Το βέλτιστο διάνυσμα είναι πάλι

$$\mathbf{w} = \sum_{k=1}^{N} \lambda_k d_k \mathbf{x}_k$$

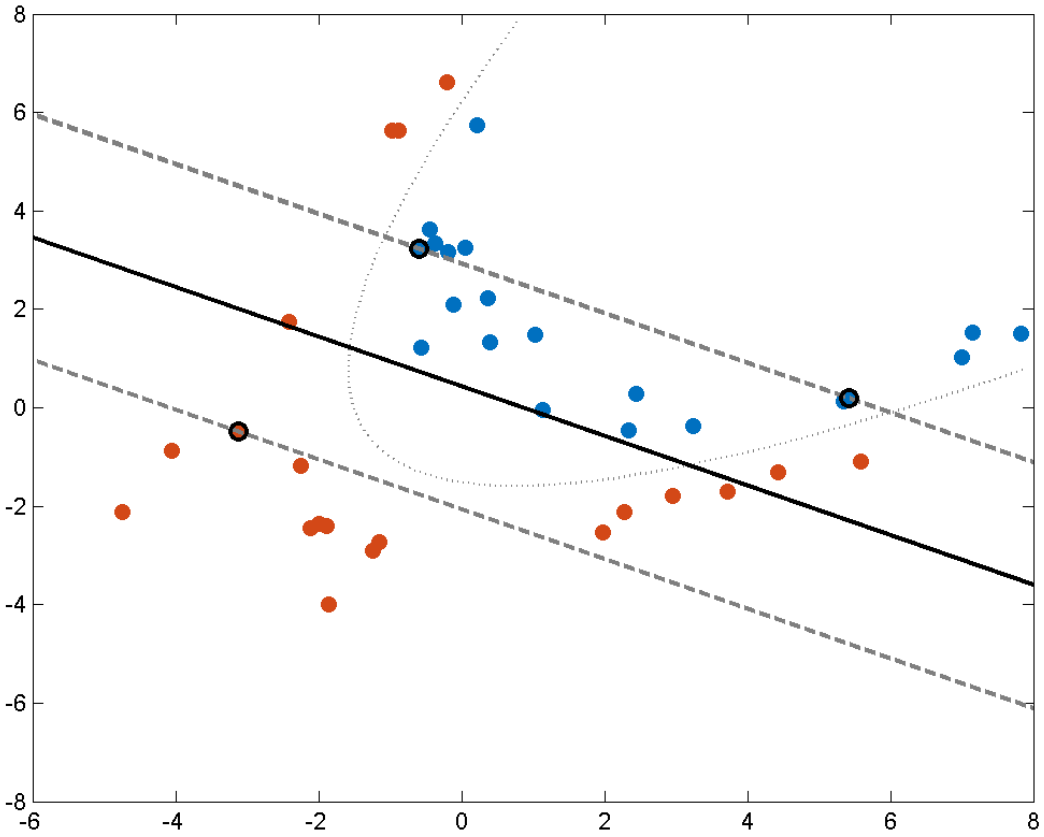
όπου στο άθροισμα συμμετέχουν ($\lambda_k > 0$) μόνο τα $\mathbf{x}_k$ για τα οποία είτε $d_k(\mathbf{w}^T\mathbf{x}_k + w_0) = 1$ (διανύσματα υποστήριξης) είτε $d_k(\mathbf{w}^T\mathbf{x}_k + w_0) < 1$

- Η βέλτιστη πόλωση είναι πάλι

$$w_0 = \frac{1}{d_k} - \mathbf{w}^T\mathbf{x}_k$$

όπου $\mathbf{x}_k$ είναι οποιοδήποτε δ.υ.

Παράδειγμα



Διαχωρισμός μη Γραμμικών κλάσεων

- Μετασχηματισμός του προβλήματος σε μεγαλύτερη διάσταση

$$\mathbf{x} \mapsto \Phi(\mathbf{x})$$

- Λύση του προβλήματος διαχωρισμού στην μεγαλύτερη διάσταση, δηλ. προσπαθούμε να βρούμε λύση στο νέο πρόβλημα:

$$d_k(\mathbf{w}^T \Phi(\mathbf{x}_k) + w_0) \geq 1, \quad k = 1, \cdots, N$$

Παράδειγμα

- Έστω ο μετασχηματισμός

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto \mathbf{z} = \Phi(\mathbf{x}) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

- Το πρόβλημα XOR …

x_1	0	0	1	1
x_2	0	1	0	1
d	-1	1	1	-1

Παράδειγμα (συν.)

- …Μετασχηματίζεται στο παρακάτω πρόβλημα

z_1	0	0	1	1
z_2	0	0	0	1.4142
z_3	0	1	0	1
d	-1	1	1	-1

- Το πρόβλημα *τώρα* είναι γραμμικά διαχωρίσιμο!!
- Δοκιμάστε πχ.

$$\mathbf{w} = [1, -\sqrt{2}, 1], \quad w_0 = -0.5$$

Πυρήνες (Kernels)

- Δυσκολία: καλούμαστε να λύσουμε το πρόβλημα σε ένα χώρο μεγαλύτερων διαστάσεων, πχ 1000 ή ακόμη και άπειρων διαστάσεων
- Ευτυχώς δε χρειάζεται να υπολογίσουμε το **w** ούτε να κάνουμε το μετασχηματισμό Φ(**x**)!
- Ας ορίσουμε τη συνάρτηση

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

- Η συνάρτηση *K*( ) καλείται πυρήνας (kernel) και η τιμή της είναι ένας αριθμός αφού είναι το εσωτερικό γινόμενο δύο διανυσμάτων Φ(**x**), Φ(**y**).

Παράδειγμα πυρήνα

- Ας θυμηθούμε το μετασχηματισμό

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto \mathbf{z} = \Phi(\mathbf{x}) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1 x_2 \\ x_2^2 \end{bmatrix}$$

- Η σχετική συνάρτηση πυρήνα είναι

$$K(\mathbf{x},\mathbf{y}) = \Phi(\mathbf{x})^T \Phi(\mathbf{y}) = (x_1^2 y_1^2 + 2x_1 x_2 y_1 y_2 + x_2^2 y_2^2)$$
$$= (x_1 y_1 + x_2 y_2)^2 = (\mathbf{x}^T \mathbf{y})^2$$

- Πιο απλός ο υπολογισμός του $(\mathbf{x}^T \mathbf{y})^2$ από τον υπολογισμό του $\Phi(\mathbf{x})^T \Phi(\mathbf{y})$

Συνηθισμένοι πυρήνες

$K(\mathbf{x},\mathbf{y}) = \exp\{-\|\mathbf{x}-\mathbf{y}\|^2/(2\sigma^2)\}$	Γκαουσιανός RBF
$K(\mathbf{x},\mathbf{y}) = [\mathbf{x}^T\mathbf{y}+\theta]^p$	Πολυωνυμικός
$K(\mathbf{x},\mathbf{y}) = \tanh(a\mathbf{x}^T\mathbf{y}+\theta)$	Σιγμοειδής
$K(\mathbf{x},\mathbf{y}) = 1/\sqrt{\|\mathbf{x}-\mathbf{y}\|^2+c^2}$	Αντίστροφος πολυτετραγωνικός

- Κάθε πυρήνας παράγεται από κάποια συνάρτηση Φ() όμως δε χρειάζεται να την ξέρουμε. Παντού στις πράξεις εμφανίζεται το *K*() και πουθενά το Φ…

Πρόβλημα Τετραγωνικού Προγραμματισμού με πυρήνες

- Σύμφωνα με το δυϊκό πρόβλημα πρέπει να ελαχιστοποιήσουμε τη συνάρτηση

$$L(\boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{\lambda}^T \mathbf{Q}\boldsymbol{\lambda} + \mathbf{g}^T\boldsymbol{\lambda}$$

- υπό τους περιορισμούς $C \geq \lambda_k \geq 0, \quad k = 1, \cdots, N$

$$\sum_{k=1}^{N} d_k \lambda_k = 0 \quad \text{ισοδ.} \quad \mathbf{d}^T\boldsymbol{\lambda} = 0$$

- αλλά τώρα

$$\mathbf{Q}: q_{ij} = d_i d_j \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) = d_i d_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\mathbf{g}: g_i = -1$$

Αρκεί η συνάρτηση πυρήνα

- Παρατηρήστε ότι οι πίνακες **Q**, **g** δεν απαιτούν γνώση της συνάρτησης Φ( ) για να υπολογιστούν. Απαιτείται μόνο γνώση της συνάρτησης πυρήνα *K*().

Λύση

- Σύμφωνα με τη θεωρία μας, τα βέλτιστα **w**, w_0 είναι

Χρειάζεται τη συνάρτηση Φ

$$\mathbf{w} = \sum_{k=1}^{N} \lambda_k d_k \Phi(\mathbf{x}_k)$$

$$w_0 = \frac{1}{d_k} - \mathbf{w}^T \Phi(\mathbf{x}_k) = \frac{1}{d_k} - \sum_{j=1}^{N} \lambda_j d_j \Phi(\mathbf{x}_j)^T \Phi(\mathbf{x}_k)$$

$$= \frac{1}{d_k} - \sum_{j=1}^{N} \lambda_j d_j K(\mathbf{x}_j, \mathbf{x}_k)$$

Δεν χρειάζεται τη συνάρτηση Φ

Δε χρειάζομαι το **w**

- Το διάνυσμα **w** δεν μας χρειάζεται. Αυτό που πραγματικά μας ενδιαφέρει είναι αν η φόρμουλα

$$y(\mathbf{x}_k) = \mathbf{w}^T \Phi(\mathbf{x}_k) + w_0 = \sum_{j=1}^{N} \lambda_j d_j K(\mathbf{x}_j, \mathbf{x}_k) + w_0$$

δίνει θετικό ή αρνητικό αποτέλεσμα

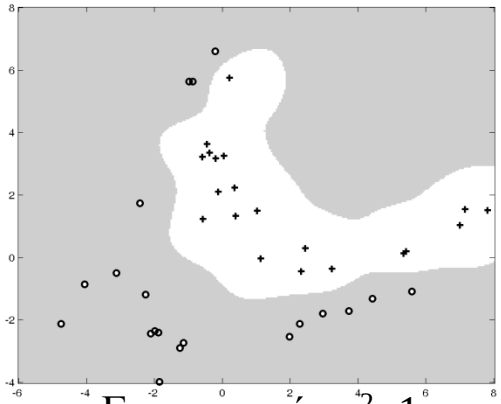
Δεν χρειάζεται τη συνάρτηση Φ

Μηχανές Διανυσμάτων Υποστήριξης

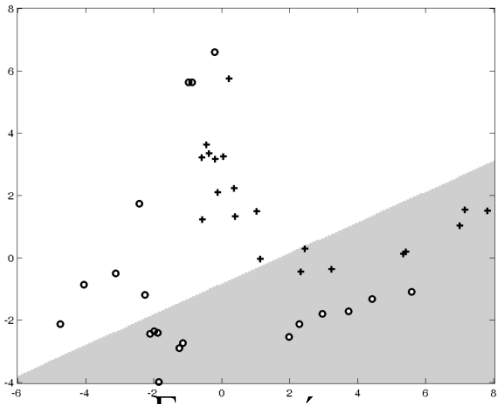
(Support Vector Machines – SVM)

- Μηχανές μάθησης που υλοποιούν την παραπάνω θεωρία με πυρήνα ή χωρίς
- Απαιτούν σαν είσοδο τα διανύσματα $\mathbf{x}_k$ μαζί με τους στόχους d_k άρα ανήκουν στην κατηγορία των μηχανών μάθησης με επίβλεψη
- Βασίζονται στην λύση προβλήματος τετραγωνικού προγραμματισμού. Το πρόβλημα έχει μελετηθεί εκτενώς στα μαθηματικά. Εκτός από τη συνάρτηση `quadprog` (MATLAB) υπάρχουν βελτιώσεις και άλλες γρήγορες υλοποιήσεις

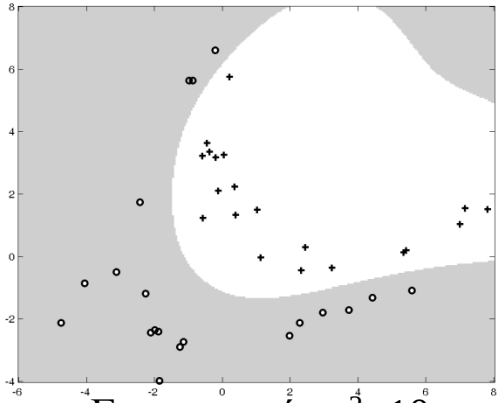
Παράδειγμα (διάφοροι πυρήνες)



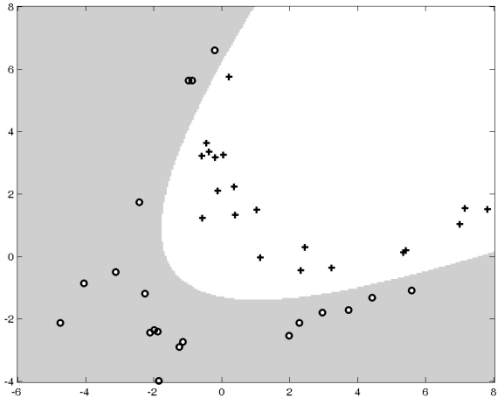
Γκαουσιανός $\sigma^2=1$



Γραμμικός



Γκαουσιανός $\sigma^2=10$



Πολυωνυμικός (τετραγωνικός)

Μέθοδοι υλοποίησης, υλικό στο διαδίκτυο

Μέθοδοι υλοποίησης

- SVM light (Torsten Joachims)

  http://svmlight.joachims.org/

- LIBSVM (Chih-Chung Chang and Chih-Jen Lin)

  http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Υλικό στο διαδίκτυο, portal

http://www.kernel-machines.org

http://www.support-vector.net

Πλεονεκτήματα και ανοικτά θέματα

Πλεονεκτήματα

- Οι Μηχανές Διανυσμάτων Υποστήριξης - ΜΔΥ (Support Vector Machines - SVM) λύνουν γραμμικά και μη γραμμικά προβλήματα διαχωρισμού
- Έχουν μεγάλη ταχύτητα υλοποίησης
- Αντιμετωπίζουν εύκολα προβλήματα μεγάλου μεγέθους

Ανοικτά προβλήματα

- Ποια είναι η βέλτιστη επιλογή πυρήνα; Κριτήρια επιλογής
- Πώς μπορώ να παραλληλοποιήσω των αλγόριθμο;

Kernel Perceptron

- Εφαρμογή της ιδέας των Kernel στον αλγόριθμο Perceptron
- Κατά την επανάληψη *k* παίρνουμε το πρότυπο $\mathbf{x}_p$
- υπολογίζουμε την έξοδο *y* με βάση τα προηγούμενα βάρη

$$y = f(\mathbf{w}(k-1)^T \Phi(\mathbf{x}_p) + w_0(k-1))$$

- υπολογίζουμε το σφάλμα $e_p = d_p - y$
- διορθώνουμε βάρη και πόλωση

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \beta\, e_p \Phi(\mathbf{x}_p)$$

$$w_0(k) = w_0(k-1) + \beta\, e_p$$

Χρήση kernel (αποφυγή Φ)

- Αποφυγή υπολογισμού του **w**. Αφού

$$\mathbf{w}(k) = \sum_{i=1}^{N} \lambda_i(k) d_i \Phi(\mathbf{x}_i)$$

- Το *y* υπολογίζεται χωρίς το **w**

$$y = f\left(\sum_{i=1}^{N} \lambda_i(k-1) K(\mathbf{x}_i, \mathbf{x}_p) + w_0(k-1)\right)$$

- Αρκεί να ενημερώσουμε τους συντελεστές $\lambda_i(k)$: στην επανάληψη *k* ενημερώνουμε μόνο το συντελεστή που αφορά το πρότυπο $\Phi(\mathbf{x}_p)$

$$\lambda_p(k) = \lambda_p(k-1) + \beta e_p$$

Kernel Perceptron

- Επέκταση του κλασικού Perceptron
- Μη γραμμικός ταξινομητής ανάλογα με τη συνάρτηση πυρήνα που χρησιμοποιείται
- Αναδρομικός κανόνας μάθησης
- Παρόμοια απόδοση με το SVM
- Πιο αργό από το SVM