Instance-based learning: Support Vector Machines

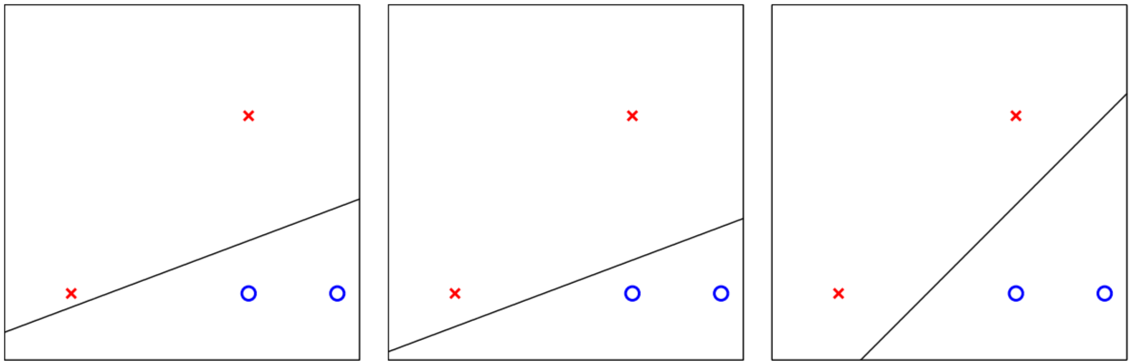
Universitat de les Illes Balears
Departament de Ciències Matemàtiques i Informàtica

11752 Aprendizaje Automático
11752 Machine Learning
Máster Universitario
en Sistemas Inteligentes

Alberto ORTIZ RODRÍGUEZ

Review on hyperplanes

- One can find several hyperplanes to separate the 2D toy dataset below:



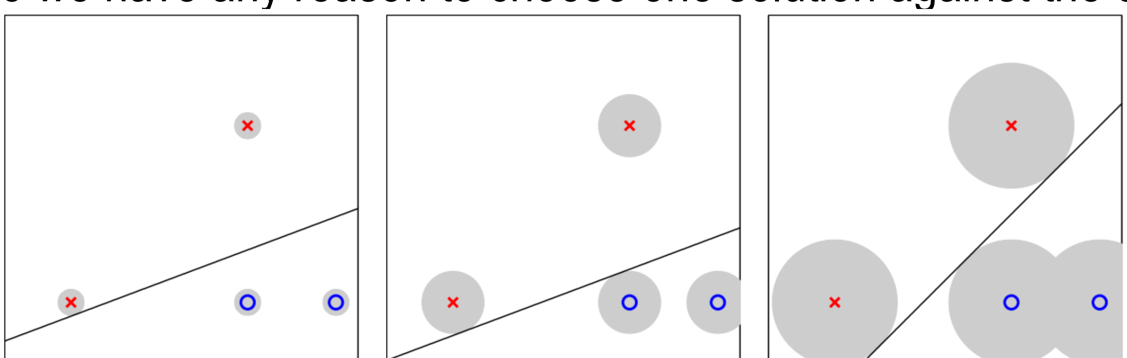
$$\frac{g(x) = w^T x + w_0}{\begin{array}{l} g(x_i) > 0 \Rightarrow x_i \to \omega_1 \\ g(x_i) < 0 \Rightarrow x_i \to \omega_2 \end{array}}$$

which could be the result of e.g. the perceptron algorithm:

$$w^+(t+1) = w^+(t) - \rho_t \sum_{x_i^+ \in \mathcal{Y}} \delta_{x_i} x_i^+, \text{ with } \delta_{x_i} = -1 \text{ if } x_i \in \omega_1, +1 \text{ if } x_i \in \omega_2$$

$$w^+ = (w, w_0),\ x_i^+ = (x_i, 1)$$

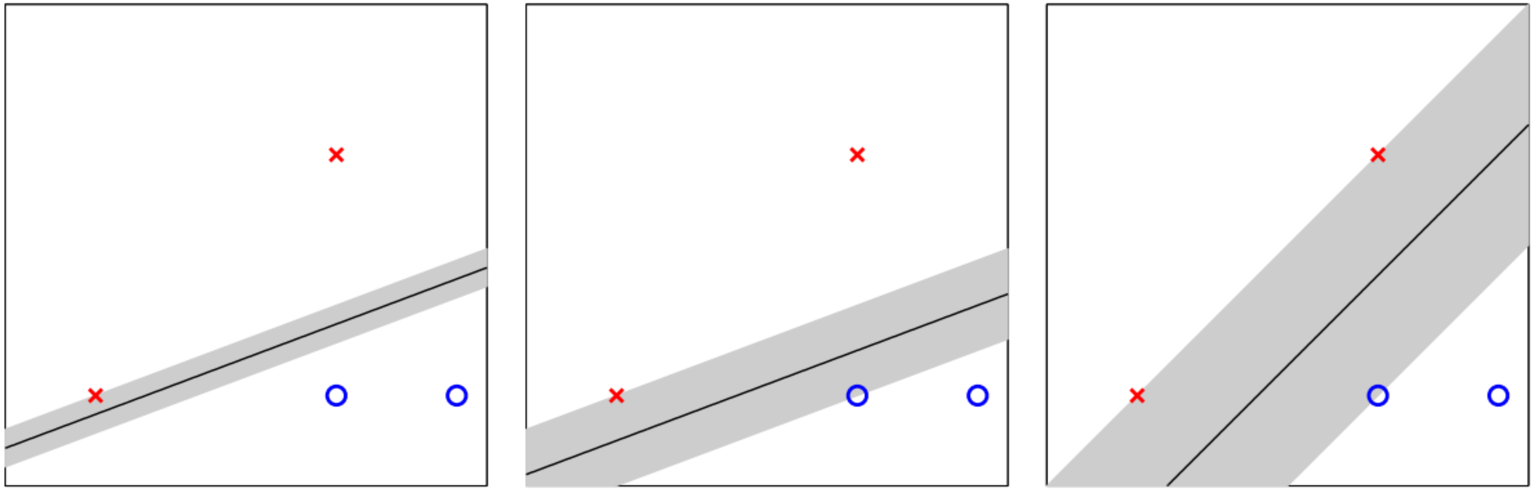
- Do we have any reason to choose one solution against the others?



$\leftarrow$ the rightmost one seems more robust to data noise, i.e. the model would keep valid even if the “true” samples were anywhere within their tolerance hypervolumes

Review on hyperplanes

- We can also quantify noise tolerance from the viewpoint of the separator, defining a "cushion" on each side of the separator, the largest one we can define:



  – We call such a "cushion" as the **margin** of the separator, so that the thicker the larger is the noise margin of the separator

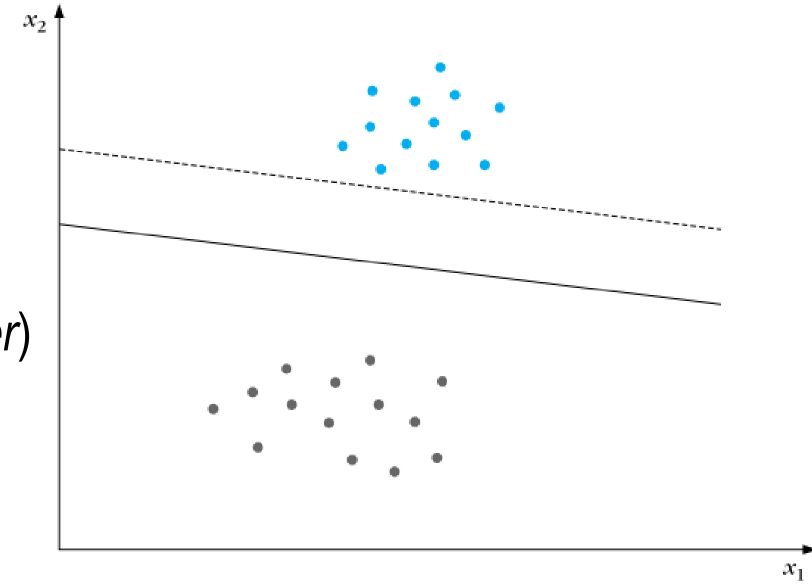
- In this lecture we will address several points in this regard:
  – Can we efficiently find the **largest margin** hyperplane?
  – What can we do if the data is **not linearly separable**?

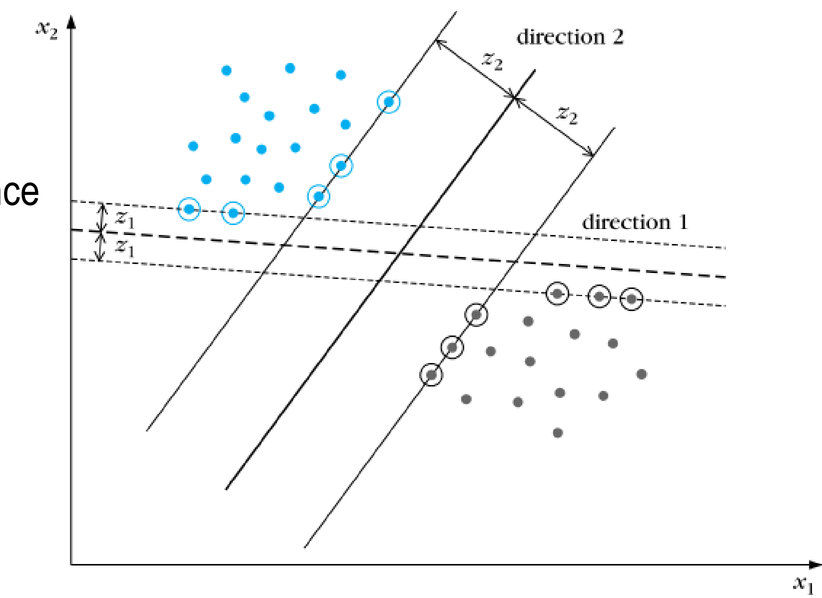
Contents

Formulation of the SVM problem

- Let $\mathbf{x}_i$, i = 1,…,N, be the feature vectors of the training set **X**, which belong to one of two **linearly separable** classes ω_1 and ω_2
- The goal is to find the separating hyperplane with the largest **margin** (*max. margin classifier*)
  - We expect that the larger the margin the **better** the **generalization** of the classifier
  - If we do not want to give preference to one class over the other, we look for the hyperplane that is at the **same orthogonal distance** to the nearest samples from ω_1 and ω_2

  $\Rightarrow$ determine the **$(\mathbf{w},w_0)$ that leads to the maximum margin**, i.e. maximum orthogonal distance
- **Support Vectors** $\equiv$ nearest samples (most informative for classification)
- **SVM** $\equiv$ optimum hyperplane





Formulation of the SVM problem

- An additional fact about classification rules based on hyperplanes, i.e. $g(x) = w^T x + w_0$

$$x = x_p + r\frac{w}{\|w\|} \Rightarrow x_p = x - r\frac{w}{\|w\|}$$

$$g(x_p) = w^T(x - r\frac{w}{\|w\|}) + w_0 = w^T x + w_0 - r\frac{w^T w}{\|w\|} = g(x) - r\|w\| = 0 \Rightarrow r = \frac{g(x)}{\|w\|}$$

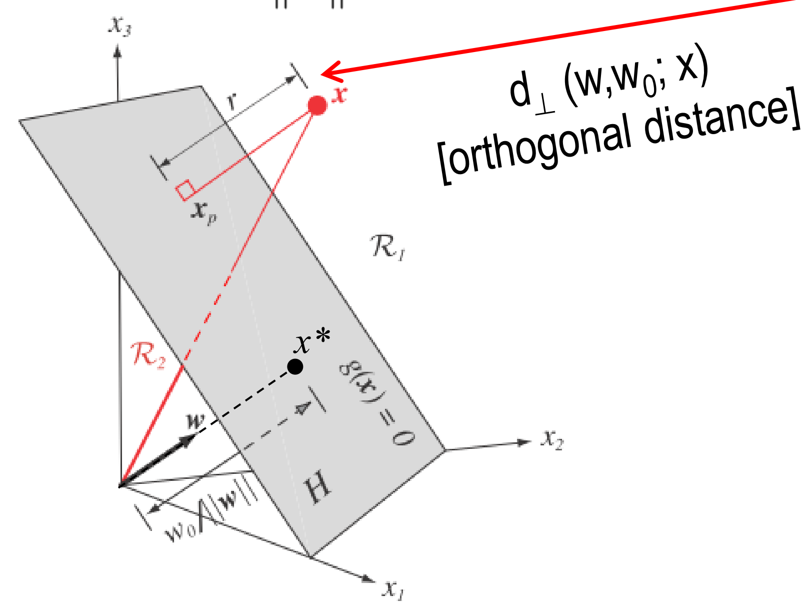


FIGURE 5.2. The linear decision boundary H, where $g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0 = 0$, separates the feature space into two half-spaces $\mathcal{R}_1$ (where $g(\mathbf{x}) > 0$) and $\mathcal{R}_2$ (where $g(\mathbf{x}) < 0$). From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Formulation of the SVM problem

- Let us define class indicators $\mathbf{y_i}$ for every sample $\mathbf{x_i}$

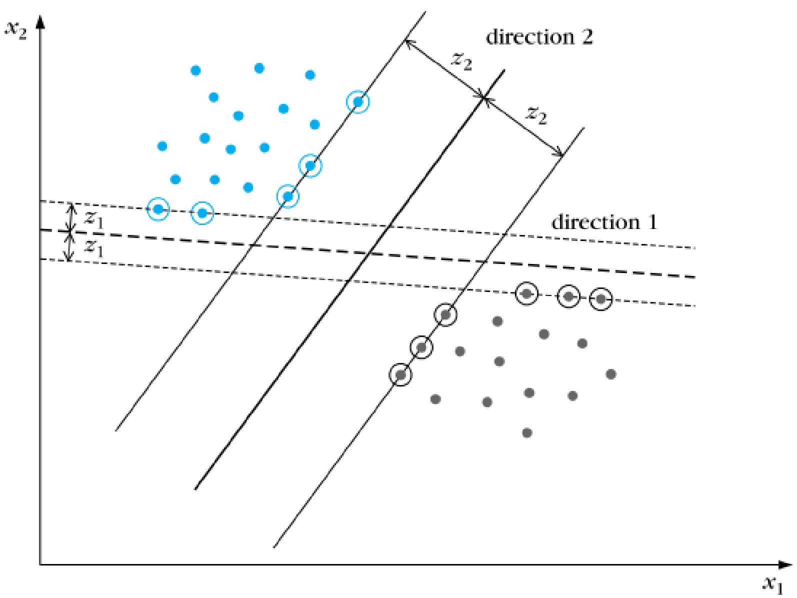
$$y_i = \begin{cases} +1 & x_i \in \omega_1 \\ -1 & x_i \in \omega_2 \end{cases} \Rightarrow \begin{array}{l} \text{search for } (w, w_0) \text{ such that} \\ y_i g(x_i) = y_i(w^T x_i + w_0) \geq 0, i = 1, \ldots, N \end{array}$$

- To solve the SVM problem, we need to maximize the margin for the $\mathbf{x_i}$'s closest to the separating hyperplane:

$$\arg\max_{w,w_0} \left\{ \min_i \ \mathrm{d}_\perp(w, w_0; x_i) \right\} \equiv \arg\max_{w,w_0} \left\{ \min_i \left[\frac{y_i(w^T x_i + w_0)}{\|w\|} \right] \right\}$$

- Let us suppose now ||w|| = 1 and that final opposite support vectors are at a distance **2z** from each other. Then:

$$y_i(w^T x_i + w_0) \geq z, i = 1, \ldots, N$$
$$\Downarrow$$
$$y_i \left(\left(\frac{w}{z} \right)^T x_i + \frac{w_0}{z} \right) \geq 1, i = 1, \ldots, N$$



Formulation of the SVM problem

- We can solve for **$w^* = w / z$** and **$w_0^* = w_0 / z = w^T x_0 / z$** and free us from the scale factor of **(w,w_0)** [$w^T(x - x_0) = w^T x + w_0 = 0 = (w^*)^T x + w_0^*$] when maximizing

$$\mathrm{d}_\perp(w, w_0; x_i) = \frac{y_i(w^T x_i + w_0)}{\|w\|} = \frac{y_i\left(\left(\frac{w}{z}\right)^T x_i + \frac{w_0}{z}\right)}{\sqrt{\left(\frac{w}{z}\right)^T \frac{w}{z}}} = \frac{y_i\left((w^*)^T x_i + w_0^*\right)}{\|w^*\|}$$

- For appropriately scaled (**w**, **w_0**), we have $\forall x_i, y_i g(x_i) = y_i(w^T x_i + w_0) \geq 1$

  and support vectors lie on hyperplanes $y_i g(x_i) = y_i(w^T x_i + w_0) = 1$

- From this, we can write $\max\limits_{w}\left\{\min\limits_{i} \frac{y_i g(x_i)}{\|w\|}\right\} = \max\limits_{w} \frac{1}{\|w\|} \equiv \min\limits_{w}\|w\|$

- According to all the aforementioned, the **SVM problem** finally becomes into a quadratic optimization problem with linear constraints/inequalities:

$$\min\ J(w) = \frac{1}{2} w^T w$$
$$\text{subject to } y_i(w^T x_i + w_0) \geq 1, i = 1, \ldots, N$$

Contents

SVM training

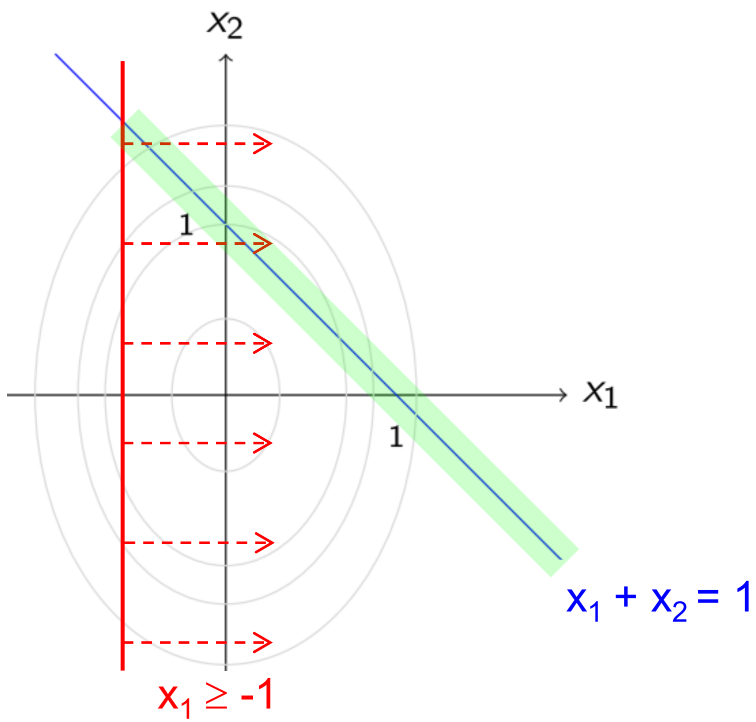
- To solve the quadratic optimization problem with linear inequality constraints

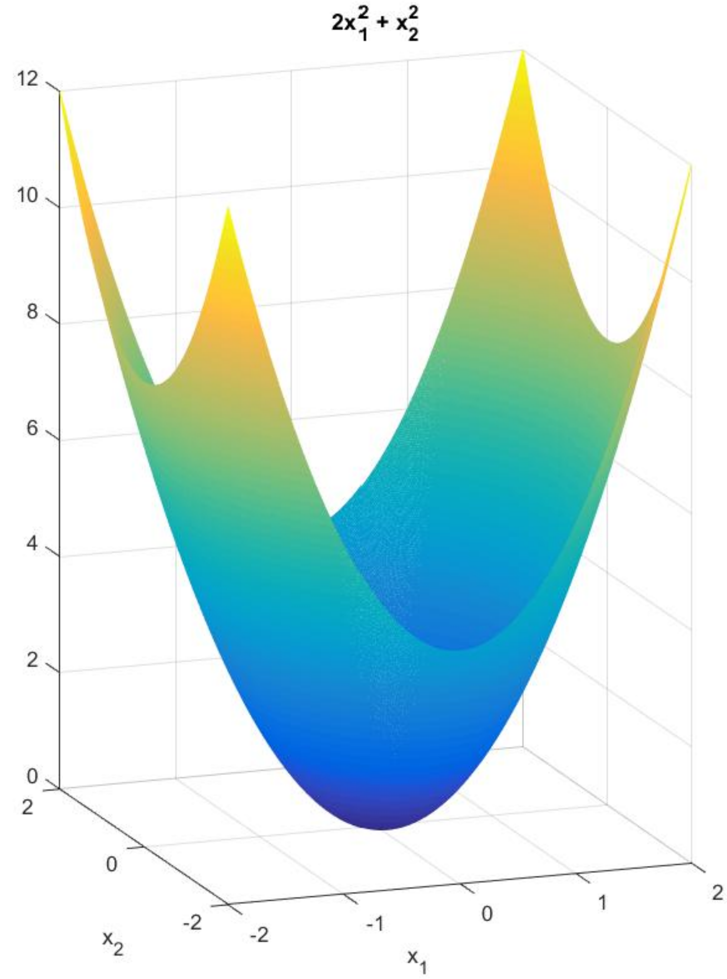
$$\min\ J(w) = \frac{1}{2} w^T w$$
$$\text{subject to } y_i(w^T x_i + w_0) \geq 1, i = 1, \ldots, N$$

  we have to resort to the **Lagrangian** function and the **Karush-Kuhn-Tucker** (KKT) conditions (necessary conditions for function extrema in problems constrained by inequalities).

- We want to solve this kind of optimization problems:

$$\begin{aligned} \min\ f(x) &= 2x_1^2 + x_2^2 \\ \text{subject to}\ x_1 + x_2 &= 1 \\ x_1 + 1 &\geq 0 \end{aligned}$$





Function optimization with constraints

- In general:

$$\textbf{min}\ f(x) \qquad\qquad [\textbf{max}\ f(x) \equiv \min\ -f(x)]$$
$$\textbf{subject to}\ g_j(x) = 0,\ j = 1, \ldots, n$$
$$h_k(x) \leq 0,\ k = 1, \ldots, m\ [h_k(x) \geq 0 \rightarrow -h_k(x) \leq 0]$$

requires the definition of the so-called **Lagrangian function**:

$$L(x, \lambda, \mu) = \quad f(x) + \sum_{j=1}^{n} \lambda_j g_j(x) + \sum_{k=1}^{m} \mu_k h_k(x) \quad [\min\ f(x)]$$
$$L(x, \lambda, \mu) = -f(x) + \sum_{j=1}^{n} \lambda_j g_j(x) + \sum_{k=1}^{m} \mu_k h_k(x) \quad [\max\ f(x)]$$

where $\{\lambda_j\}$ and $\{\mu_k\}$ are the **Karush-Kuhn-Tucker multipliers** (Lagrange multipliers if there are no inequalities)

- The solution to the optimization problem is among the solutions of the **KKT conditions**

$$(1)\ \frac{\partial L}{\partial x_i} = 0,\ (2)\ \frac{\partial L}{\partial \lambda_j} = 0,\ (3)\ \mu_k h_k(x) = 0,\ (4)\ \mu_k \geq 0$$

  – They are **necessary conditions** for locating function extrema in problems constrained by equalities and/or inequalities

Function optimization with constraints

- **Example**:

$$\begin{aligned} \mathbf{min}\ & f(x) = 2x_1^2 + x_2^2 \\ \mathbf{subject\ to}\ & x_1 + x_2 = 1 \\ & x_1 + 1 \geq 0 \end{aligned}$$

$$\begin{aligned} \mathbf{min}\ & f(x) = 2x_1^2 + x_2^2 \\ \mathbf{subject\ to}\ & g(x) = x_1 + x_2 - 1 = 0 \\ & h(x) = -(x_1 + 1) \leq 0 \end{aligned}$$

$$L(x, \lambda, \mu) = f(x) + \sum_{j=1}^{n} \lambda_i g_j(x) + \sum_{k=1}^{m} \mu_k h_k(x)$$

$$\downarrow$$

$$L(x, \lambda, \mu) = 2x_1^2 + x_2^2 + \lambda(x_1 + x_2 - 1) + \mu(-x_1 - 1)$$

$$\frac{\partial L}{\partial x_1} = 4x_1 + \lambda - \mu = 0$$

$$\frac{\partial L}{\partial x_2} = 2x_2 + \lambda = 0$$

$$\frac{\partial L}{\partial \lambda} = x_1 + x_2 - 1 = 0$$

$$\mu(-(x_1 + 1)) = 0, \mu \geq 0$$

$$\boxed{\begin{aligned} & \mu = 0 \\ & 4x_1 + \lambda = 0 \Rightarrow x_1 = -\lambda/4 = 1/3 \\ & 2x_2 + \lambda = 0 \Rightarrow x_2 = -\lambda/2 = 2/3 \\ & x_1 + x_2 - 1 = 0 \Rightarrow \lambda = -4/3 \\ & \mu(-(x_1 + 1)) = 0, \mu \geq 0 \end{aligned}}$$

$$\boxed{\begin{aligned} & x_1 + 1 = 0 \Rightarrow x_1 = -1 \\ & -4 + \lambda - \mu = 0 \Rightarrow \mu = -8 \\ & 2x_2 + \lambda = 0 \Rightarrow \lambda = -4 \\ & -1 + x_2 - 1 = 0 \Rightarrow x_2 = 2 \\ & \mu = -8 \not\geq 0,\ \text{NOT a solution} \end{aligned}}$$

SVM training

- A **first** solution to the quadratic optimization problem associated to SVM training

$$\min \; J(w) = \frac{1}{2} w^T w$$
$$\text{subject to } y_i (w^T x_i + w_0) \geq 1, i = 1, \ldots, N$$

PRIMAL PROBLEM

is obtained by means of the corresponding **Lagrangian** function

$$L(w, w_0, \lambda) = \tfrac{1}{2} w^T w - \sum_{i=1}^{N} \lambda_i \left[y_i (w^T x_i + w_0) - 1 \right]$$

and the **Karush-Kuhn-Tucker** (KKT) conditions:

$$\begin{aligned} &\frac{\partial L}{\partial w} = 0 \\ &\frac{\partial L}{\partial w_0} = 0 \\ &\lambda_i \left[y_i (w^T x_i + w_0) - 1 \right] = 0, i = 1, \ldots, N \\ &\lambda_i \geq 0, i = 1, 2, \ldots, N \end{aligned} \Rightarrow \begin{cases} w - \sum_{i=1}^{N} \lambda_i y_i x_i = 0 \Rightarrow w = \sum_{i=1}^{N} \lambda_i y_i x_i \\ \sum_{i=1}^{N} \lambda_i y_i = 0 \\ \lambda_i \left[y_i (w^T x_i + w_0) - 1 \right] = 0, i = 1, \ldots, N \\ \lambda_i \geq 0, i = 1, \ldots, N \end{cases}$$

SVM training

- **Remarks**:
  1) **w** is a linear combination of the feature vectors for which $\boldsymbol{\lambda_i} \neq 0$:

     $$w = \sum_{i=1}^{N} \lambda_i y_i x_i = \sum_{i|\lambda_i \neq 0} \lambda_i y_i x_i$$

  2) Regarding $\lambda_i [y_i(w^T x_i + w_0) - 1] = 0$, when $\boldsymbol{\lambda_i} \neq 0$, the corresponding constraint is called **active**, and makes the corresponding $\mathbf{x_i}$ lie on either of the two hyperplanes $w^T x_i + w_0 = \pm 1$.

     $\mathbf{x_i}$ such that $\boldsymbol{\lambda_i} \neq 0$ are, thus, the **support vectors** and constitute the critical elements of the training set.

     Feature vectors corresponding to $\boldsymbol{\lambda_i} = 0$ can either lie outside the **class separation band**, defined as the region between the two hyperplanes, or they can also lie on one of these hyperplanes (degenerate cases).

  3) The resulting hyperplane is **insensitive to the number and position of the non-support vectors**, provided they do not cross the class separation band.

- **Remarks**:
  4) $\mathbf{w_0}$ can be deduced from the active constraints:

$$\lambda_i \left[y_i(w^T x_i + w_0) - 1 \right] = 0 \overset{\lambda_i \neq 0}{\Rightarrow} y_i(w^T x_i + w_0) = 1$$
$$\Rightarrow w^T x_i + w_0 = \frac{1}{y_i} = y_i \ (\text{since } y_i = \pm 1)$$
$$\Rightarrow w_0 = y_i - w^T x_i$$
$$\Rightarrow w_0 = y_i - \left(\sum_{j|\lambda_j \neq 0} \lambda_j y_j x_j^T \right) x_i$$

  In practice, $\mathbf{w_0}$ is computed as an average value obtained from all $\mathbf{N_\lambda}$ active constraints (it is numerically safer):

$$w_0 = \frac{1}{N_\lambda} \sum_{i|\lambda_i \neq 0} \left(y_i - \left(\sum_{j|\lambda_j \neq 0} \lambda_j y_j x_j^T \right) x_i \right)$$
$$= \frac{1}{N_\lambda} \sum_{i|\lambda_i \neq 0} y_i - \frac{1}{N_\lambda} \sum_{i,j|\lambda_i,\lambda_j \neq 0} \lambda_j y_j x_j^T x_i$$

  5) Due to the nature of the cost function (convex) and the constraints (linear), the SVM is guaranteed to be **unique**.

- We have yet to determine the λ_i. To this end, **w** and **w₀** are substituted in the Lagrangian using the equality constraints from the 1st solution (**Wolfe dual repres**.)

$$\left.\begin{array}{c} L(w, w_0, \lambda) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \lambda_i \left[y_i (w^T x_i + w_0) - 1 \right] \\ \boxed{w = \sum_{i=1}^{N} \lambda_i y_i x_i}, \quad \boxed{\sum_{i=1}^{N} \lambda_i y_i = 0} \end{array}\right\} \Rightarrow$$

$$L(\lambda) = \boxed{\frac{1}{2} \left(\sum_{i=1}^{N} \lambda_i y_i x_i \right)^T \left(\sum_{j=1}^{N} \lambda_j y_j x_j \right) - \sum_{i=1}^{N} \lambda_i y_i \left(\sum_{j=1}^{N} \lambda_j y_j x_j \right)^T x_i} - w_0 \underbrace{\sum_{i=1}^{N} \lambda_i y_i}_{0} + \sum_{i=1}^{N} \lambda_i$$

$$= \sum_{i=1}^{N} \lambda_i \boxed{- \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j}$$

- The optimization problem becomes again into a **quadratic optimization problem**, to solve for λ_i

$$\max L(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j$$

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

$$\lambda_i \geq 0, i = 1, \ldots, N$$

DUAL PROBLEM

- Given: $$\max L(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j$$

$$\sum_{i=1}^{N} \lambda_i y_i = 0, \;\; \lambda_i \geq 0, i = 1, \ldots, N$$

DUAL PROBLEM

the solution by means of the KKT conditions turns out to be:

$$L(\lambda_i, \mu, \delta_i) = \overbrace{\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j - \sum_{i=1}^{N} \lambda_i}^{-f(x)} + \mu \sum_{i=1}^{N} \lambda_i y_i - \overbrace{\sum_{i=1}^{N} \delta_i \lambda_i}^{-\lambda_i \leq 0}$$

$$\frac{\partial L}{\partial \lambda_i} = \sum_{j=1}^{N} \lambda_j y_i y_j x_i^T x_j - 1 + \mu y_i - \delta_i = 0$$

$$\frac{\partial L}{\partial \mu} = \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$\delta_i \lambda_i = 0, \;\; \delta_i \geq 0, \;\; i = 1, \ldots, N$$

- In matrix form, we can write:

$$\frac{\partial L}{\partial \Lambda} = 0 \Rightarrow H\Lambda + \mu Y - \Delta = \mathbf{1} \quad \Lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \Delta = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_N \end{bmatrix}, \mathbf{1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

$$\frac{\partial L}{\partial \mu} = 0 \Rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$\delta_i \lambda_i = 0,\ \delta_i \geq 0,\ i = 1, \ldots, N \qquad H = \begin{bmatrix} y_1 y_1 x_1^T x_1 & y_1 y_2 x_1^T x_2 & \ldots & y_1 y_N x_1^T x_N \\ y_2 y_1 x_2^T x_1 & y_2 y_2 x_2^T x_2 & \ldots & y_2 y_N x_2^T x_N \\ \vdots & \vdots & \ddots & \vdots \\ y_N y_1 x_N^T x_1 & y_N y_2 x_N^T x_2 & \ldots & y_N y_N x_N^T x_N \end{bmatrix}$$

- Although the hyperplane is unique, there is no guarantee of the uniqueness of the associated Lagrange multipliers λ_i and by extension of the expansion of **w** in terms of support vectors
- Because of the size of this problem when **N** is large, a number of efficient solutions have been developed (e.g. Platt's **Sequential Minimal Optimization** – SMO)

SVM training

- **SVM algorithm**:
  - Solve for the λ_i, i = 1, …, N

$$H\Lambda + \mu Y - \Delta = \mathbf{1}$$

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

$$\delta_i \lambda_i = 0,\ \ \delta_i \geq 0,\ \ i = 1, \ldots, N$$

$$\Lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \Delta = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_N \end{bmatrix}, \mathbf{1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

$$H = \begin{bmatrix} y_1 y_1 x_1^T x_1 & y_1 y_2 x_1^T x_2 & \ldots & y_1 y_N x_1^T x_N \\ y_2 y_1 x_2^T x_1 & y_2 y_2 x_2^T x_2 & \ldots & y_2 y_N x_2^T x_N \\ \vdots & \vdots & \ddots & \vdots \\ y_N y_1 x_N^T x_1 & y_N y_2 x_N^T x_2 & \ldots & y_N y_N x_N^T x_N \end{bmatrix}$$

  - Solve for **w**:

$$w = \sum_{i|\lambda_i \neq 0} \lambda_i y_i x_i$$

  - Solve for $\mathbf{w_0}$:

$$w_0 = \frac{1}{N_\lambda} \sum_{i|\lambda_i \neq 0} \left(y_i - w^T x_i \right)$$

- An even **higher-level** view:

$$\min \quad J(w, w_0) = \frac{1}{2} w^T w$$
$$\text{s.t.} \quad y_i(w^T x_i + w_0) \geq 1, \; i = 1, \ldots, N$$

Wolfe dual representation

$$\max \; L(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j$$
$$\text{s.t.} \; \textstyle\sum_{i=1}^{N} \lambda_i y_i = 0$$
$$\lambda_i \geq 0, \; i = 1, \ldots, N$$

(1) solve for λ

(2) $w = \displaystyle\sum_{i|\lambda_i \neq 0} \lambda_i y_i x_i$

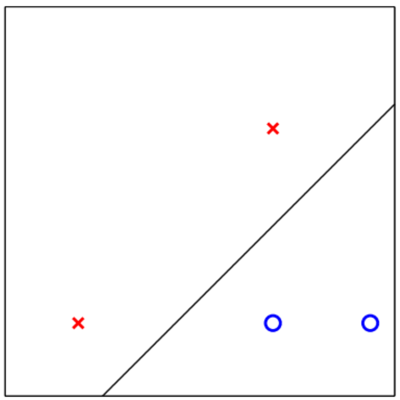
(3) $w_0 = \displaystyle\frac{1}{N_\lambda} \sum_{i|\lambda_i \neq 0} y_i - \sum_{i,j|\lambda_i,\lambda_j \neq 0} \lambda_j y_j x_j^T x_i$

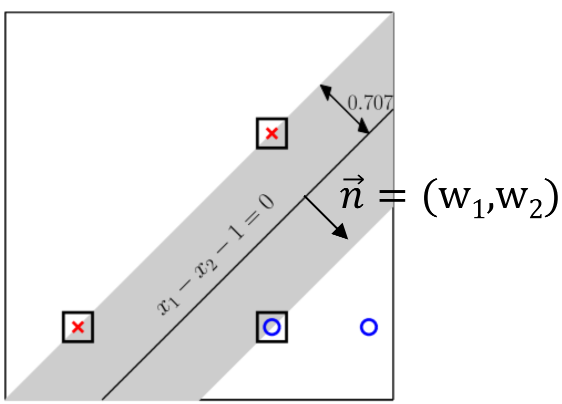
classify: $\text{sign}(w^T x + w_0) \equiv$

$$\text{sign}\left(\sum_{i|\lambda_i \neq 0} \lambda_i y_i x_i^T x + w_0 \right)$$

- Example 1(a)

$$X = \begin{bmatrix} 0 & 0 \\ 2 & 2 \\ 2 & 0 \\ 3 & 0 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$





$$\min\ J(w) = \frac{1}{2} w^T w = \frac{1}{2}(w_1^2 + w_2^2)$$

$$\text{subject to } y_i(w^T x_i + w_0) \geq 1, i = 1, \ldots, N$$

$$i) -1\,(0 \cdot w_1 + 0 \cdot w_2 + w_0) \geq 1 \Rightarrow -w_0 \geq 1$$
$$ii) -1\,(2 \cdot w_1 + 2 \cdot w_2 + w_0) \geq 1 \Rightarrow -(2w_1 + 2w_2 + w_0) \geq 1$$
$$iii) +1\,(2 \cdot w_1 + 0 \cdot w_2 + w_0) \geq 1 \Rightarrow 2w_1 + w_0 \geq 1$$
$$iv) +1\,(3 \cdot w_1 + 0 \cdot w_2 + w_0) \geq 1 \Rightarrow 3w_1 + w_0 \geq 1$$

$$i) : w_0 \leq -1$$
$$i) \text{ and } iii) : 2w_1 - 1 \geq 2w_1 + w_0 \geq 1 \Rightarrow w_1 \geq 1$$
$$ii) \text{ and } iii) : 1 + 2w_2 \leq 2w_1 + 2w_2 + w0 \leq -1 \Rightarrow w_2 \leq -1$$

$$\Rightarrow \min\ J(=1) \text{ for } w_1 = 1 \text{ and } w_2 = -1$$

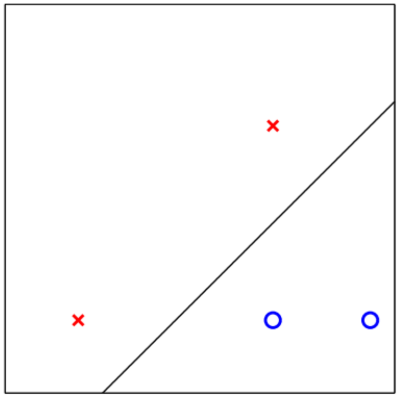
$$\forall \text{ support vector } x_i,\ y_i(w^T x_i + w_0) = 1$$
$$w_0 : -(0 \cdot w_1 + 0 \cdot w_2 + w_0) = 1 \Rightarrow w_0 = -1$$
$$: -(2 \cdot w_1 + 2 \cdot w_2 + w_0) = 1 \Rightarrow w_0 = -1$$
$$: +(2 \cdot w_1 + 0 \cdot w_2 + w_0) = 1 \Rightarrow w_0 = -1$$

$$\text{hyperplane: } (w_1 = 1, w_2 = -1, w_0 = -1) \rightarrow x_1 - x_2 - 1 = 0$$

$$\text{margin: } 1/\|w\| = 0.7071$$

- <u>Example 1</u>(b)

$$X = \begin{bmatrix} 0 & 0 \\ 2 & 2 \\ 2 & 0 \\ 3 & 0 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$



$$\max\ L(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j$$

$$\text{s.t. } \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$\lambda_i \geq 0,\ i = 1, \ldots, N$$

$$H\Lambda + \mu Y - \Delta = \mathbf{1}$$

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

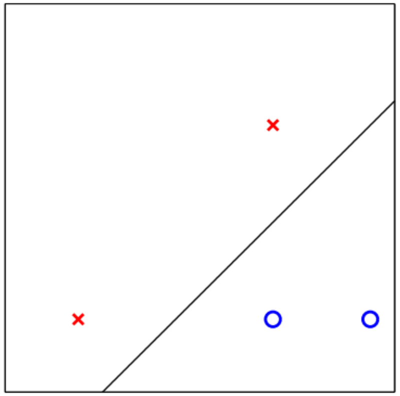
$$\delta_i \lambda_i = 0,\ \delta_i \geq 0,\ i = 1, \ldots, N$$

$$\Lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \Delta = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_N \end{bmatrix}, \mathbf{1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

$$H = \begin{bmatrix} y_1 y_1 x_1^T x_1 & y_1 y_2 x_1^T x_2 & \ldots & y_1 y_N x_1^T x_N \\ y_2 y_1 x_2^T x_1 & y_2 y_2 x_2^T x_2 & \ldots & y_2 y_N x_2^T x_N \\ \vdots & \vdots & \ddots & \vdots \\ y_N y_1 x_N^T x_1 & y_N y_2 x_N^T x_2 & \ldots & y_N y_N x_N^T x_N \end{bmatrix}$$

- Example 1(b)

$$X = \begin{bmatrix} 0 & 0 \\ 2 & 2 \\ 2 & 0 \\ 3 & 0 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$



$$H\Lambda + \mu Y - \Delta = \mathbf{1}$$

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

$$\delta_i \lambda_i = 0,\ \delta_i \geq 0,\ i = 1, \ldots, N$$

$$\Lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \Delta = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_N \end{bmatrix}, \mathbf{1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

$$H = \begin{bmatrix} y_1 y_1 x_1^T x_1 & y_1 y_2 x_1^T x_2 & \ldots & y_1 y_N x_1^T x_N \\ y_2 y_1 x_2^T x_1 & y_2 y_2 x_2^T x_2 & \ldots & y_2 y_N x_2^T x_N \\ \vdots & \vdots & \ddots & \vdots \\ y_N y_1 x_N^T x_1 & y_N y_2 x_N^T x_2 & \ldots & y_N y_N x_N^T x_N \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & +8 & -4 & -6 \\ 0 & -4 & +4 & +6 \\ 0 & -6 & +6 & +9 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix} + \mu \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix} - \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$-\lambda_1 - \lambda_2 + \lambda_3 + \lambda_4 = 0$$

$$\delta_1\, \lambda_1 = 0, \delta_2\, \lambda_2 = 0, \delta_3\, \lambda_3 = 0, \delta_4\, \lambda_4 = 0$$

$$\delta_1 \geq 0, \delta_2 \geq 0, \delta_3 \geq 0, \delta_4 \geq 0$$

p.e. $\delta_1 = \delta_2 = \delta_3 = 0$ and $\delta_4 > 0 \Rightarrow \lambda_4 = 0$

$-\mu = 1 \Rightarrow \mu = -1$

$8\lambda_2 - 4\lambda_3 - \mu = 1 \Rightarrow \lambda_3 = 2\lambda_2$

$-4\lambda_2 + 4\lambda_3 + \mu = 1 \Rightarrow -4\lambda_2 + 8\lambda_2 = 2 \Rightarrow \lambda_2 = 0.5, \lambda_3 = 1$

$-6\lambda_2 + 6\lambda_3 + \mu - \delta_4 = 1 \Rightarrow \delta_4 = 1$

$-\lambda_1 - \lambda_2 + \lambda_3 = 0 \Rightarrow \lambda_1 = \lambda_3 - \lambda_2 = 0.5$

$\Rightarrow L = 1$

- <u>Example 1</u>(b)

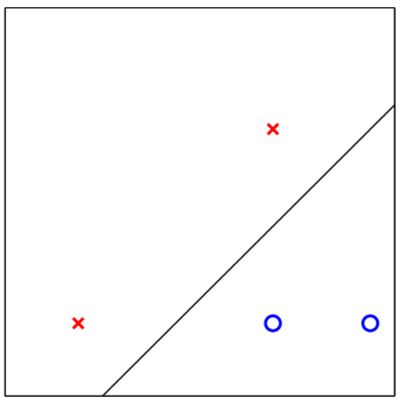
$$\mathrm{X} = \begin{bmatrix} 0 & 0 \\ 2 & 2 \\ 2 & 0 \\ 3 & 0 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$

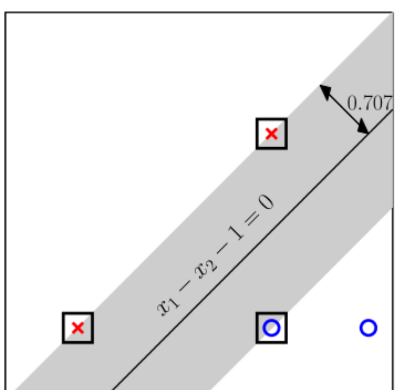
$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & +8 & -4 & -6 \\ 0 & -4 & +4 & +6 \\ 0 & -6 & +6 & +9 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix} + \mu \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix} - \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$-\lambda_1 - \lambda_2 + \lambda_3 + \lambda_4 = 0$$

$$\delta_1 \lambda_1 = 0, \delta_2 \lambda_2 = 0, \delta_3 \lambda_3 = 0, \delta_4 \lambda_4 = 0$$

$$\delta_1 \geq 0, \delta_2 \geq 0, \delta_3 \geq 0, \delta_4 \geq 0$$





δ_1	δ_2	δ_3	δ_4	λ_1	λ_2	λ_3	λ_4	μ	L
+0.00	+0.00	+0.00	+0.00	–	–	–	–	–	–
+0.00	+0.00	+0.00	+1.00	+0.50	+0.50	+1.00	+0.00	-1.00	+1.00
+0.00	+0.00	-0.67	+0.00	+0.11	+0.33	+0.00	+0.44	-1.00	–
+0.00	+0.00	-2.00	-2.00	+0.00	+0.00	+0.00	+0.00	-1.00	–
+0.00	–	+0.00	+0.00	–	+0.00	–	–	–	–
+0.00	-2.00	+0.00	+1.00	+0.50	+0.00	+0.50	+0.00	-1.00	–
+0.00	-1.33	-0.67	+0.00	+0.22	+0.00	+0.00	+0.22	-1.00	–
+0.00	+0.00	-2.00	-2.00	+0.00	+0.00	+0.00	+0.00	-1.00	–
-2.00	+0.00	+0.00	+0.00	+0.00	+0.50	+0.50	+0.00	+1.00	–
-2.00	+0.00	+0.00	+0.00	+0.00	+0.50	+0.50	+0.00	+1.00	–
-0.80	+0.00	-0.40	+0.00	+0.00	+0.40	+0.00	+0.40	-0.20	–
+0.00	+0.00	-2.00	-2.00	+0.00	+0.00	+0.00	+0.00	-1.00	–
-2.00	-2.00	+0.00	+0.00	+0.00	+0.00	+0.00	+0.00	+1.00	–
-2.00	-2.00	+0.00	+0.00	+0.00	+0.00	+0.00	+0.00	+1.00	–
-2.00	-2.00	+0.00	+0.00	+0.00	+0.00	+0.00	+0.00	+1.00	–
-1.00	-1.00	-1.00	-1.00	+0.00	+0.00	+0.00	+0.00	+0.00	–

$$w = \sum_{i|\lambda_i \neq 0} \lambda_i y_i x_i = 0.5(-1)\begin{bmatrix} 0 \\ 0 \end{bmatrix} + 0.5(-1)\begin{bmatrix} 2 \\ 2 \end{bmatrix} + 1(+1)\begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$w_0 = \frac{1}{N_\lambda} \sum_{i|\lambda_i \neq 0} (y_i - w^T x_i) = \frac{-1-1-1}{3} = -1$$

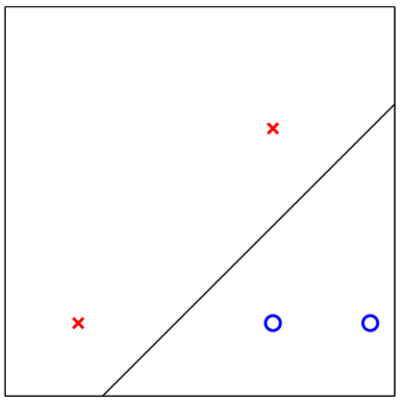
- <u>Example 1</u>(c)

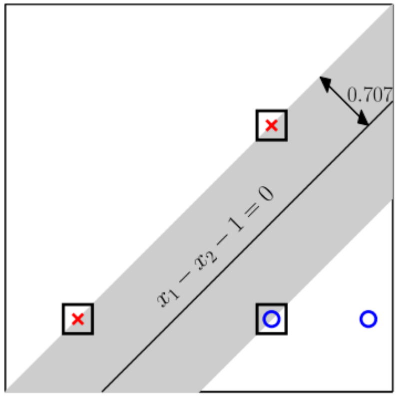
$$\max\ L(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j$$

$$\text{X} = \begin{bmatrix} 0 & 0 \\ 2 & 2 \\ 2 & 0 \\ 3 & 0 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$

$$\text{s.t. } \sum_{i=1}^{N} \lambda_i y_i = 0$$
$$\lambda_i \geq 0,\ i = 1, \ldots, N$$

Using a QP solver, e.g. **cvxpy**:
pip install cvxpy
or *conda install -c conda-forge cvxpy*





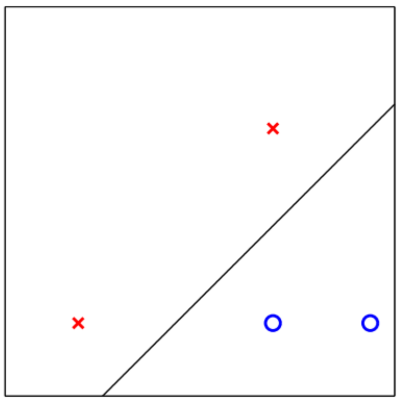

```python
import cvxpy as cp
X = np.array([[0.,0.],[2.,2.],[2.,0.],[3.,0.]])
N = X.shape[0]
y = np.array([-1.,-1.,1.,1.]).reshape((N,1))
P = build_H(X,y)
G = np.identity(N)
h = np.zeros((N,1))
A = y.reshape((1,N))
b = 0.0
z = cp.Variable((N,1))
P = P + (1e-8) * np.identity(N) # for numerical stability
prob = cp.Problem(cp.Maximize(cp.sum(z) - 0.5*cp.quad_form(z,P)),
                  [G @ z >= h, A @ z == b])
prob.solve()
lm = z.value # lm = [0.5, 0.5, 1.0, 0.0]
```

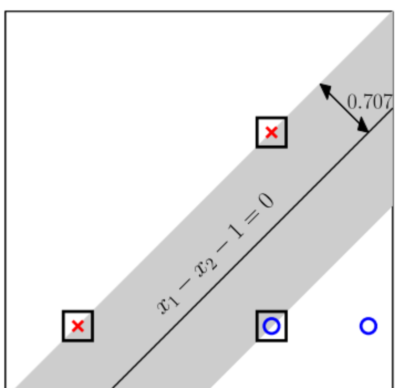
$$w = \sum_{i|\lambda_i \neq 0} \lambda_i y_i x_i = 0.5(-1)\begin{bmatrix}0\\0\end{bmatrix} + 0.5(-1)\begin{bmatrix}2\\2\end{bmatrix} + 1(+1)\begin{bmatrix}2\\0\end{bmatrix} = \begin{bmatrix}1\\-1\end{bmatrix}$$

$$w_0 = \frac{1}{N_\lambda}\sum_{i|\lambda_i \neq 0}(y_i - w^T x_i) = \frac{-1-1-1}{3} = -1$$

- <u>Example 1</u>(d)

$$X = \begin{bmatrix} 0 & 0 \\ 2 & 2 \\ 2 & 0 \\ 3 & 0 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$





$$\max L(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j$$

$$\text{s.t. } \sum_{i=1}^{N} \lambda_i y_i = 0$$
$$\lambda_i \geq 0, \; i = 1, \ldots, N$$

<u>Solve the primal problem</u>

$$\min J(w) = \frac{1}{2} w^T w$$

$$\text{subject to } y_i(w^T x_i + w_0) \geq 1, \forall i$$

Using a QP solver, e.g. **cvxpy**:

pip install cvxpy

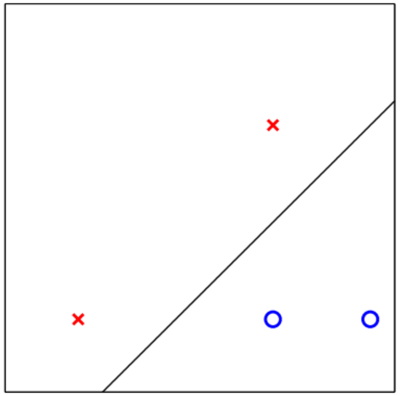
or *conda install -c conda-forge cvxpy*

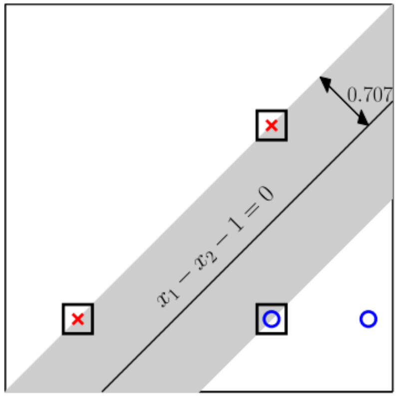
```
import cvxpy as cp
X = np.array([[0.,0.],[2.,2.],[2.,0.],[3.,0.]])
N = X.shape[0]
y = np.array([-1.,-1.,1.,1.]).reshape((N,1))
w = cp.Variable((2,1))
w0 = cp.Variable()
loss = cp.Minimize(0.5 * cp.square(cp.norm(w)))
constr = []
for i in range(N):
    xi, yi = X[i,:], y[i]
    constr += [yi @ (xi @ w + w0) >= 1]
prob = cp.Problem(loss, constr)
prob.solve()
print(w.value, w0.value) # w = [1.0, -1.0], w0 = -1.0
```

care with this formulation, since one does not have access to the λ's

- Example 1(e)

$$X = \begin{bmatrix} 0 & 0 \\ 2 & 2 \\ 2 & 0 \\ 3 & 0 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$





$$\min \quad J(w, w_0) = \frac{1}{2} w^T w$$
$$\text{s.t.} \quad y_i(w^T x_i + w_0) \geq 1,\; i = 1, \ldots, N$$

Using **scikit-learn**:

```
from sklearn import svm

X = np.array([[0.,0.],[2.,2.],[2.,0.],[3.,0.]])
N = X.shape[0]
y = np.array([-1.,-1.,1.,1.]).reshape((N,1))
clf = svm.SVC(C = 1e16, kernel = 'linear')
clf.fit(X, y)
sv = clf.support_vectors_
w = clf.coef_.flatten()
w0 = clf.intercept_
lm = clf.dual_coeff_.flatten()

# sv = [[0.,0.], [2.,2.], [2.,0.]]
# w = [1.0, -1.0], w0 = -1.0
# lm = [-0.5, -0.5, 1] # y_i * lambda_i
```

- **M-class problems**
  1) Transform it into $\boldsymbol{M}$ two-class problems (*one-versus-rest* [**OVR**], *one-versus-all* [OVA])

$$g_i(x), i = 1, \ldots, M \mid g_i(x) > 0 \text{ if } x \in \omega_i \text{ and } g_i(x) < 0 \text{ if } x \notin \omega_i$$

  - It is an unbalanced problem since the negative class can comprise far more samples than the positive class

  2) Transform it into $\boldsymbol{M(M-1)/2}$ two-class problems (*one-versus-one* [**OVO**])

$$g_{ij}(x),\ i, j = 1, \ldots, M, i \neq j \mid g_{ij}(x) > 0 \text{ if } x \in \omega_i$$

$g_{12}(x)$	$g_{13}(x)$	$g_{23}(x)$	class
< 0 ω_2	< 0 ω_3	< 0 ω_3	$\rightarrow \omega_3$
< 0 ω_2	< 0 ω_3	> 0 ω_2	$\rightarrow \omega_2$
< 0 ω_2	> 0 ω_1	< 0 ω_3	?
< 0 ω_2	> 0 ω_1	> 0 ω_2	$\rightarrow \omega_2$

$g_{12}(x)$	$g_{13}(x)$	$g_{23}(x)$	class
> 0 ω_1	< 0 ω_3	< 0 ω_3	$\rightarrow \omega_3$
> 0 ω_1	< 0 ω_3	> 0 ω_2	?
> 0 ω_1	> 0 ω_1	< 0 ω_3	$\rightarrow \omega_1$
> 0 ω_1	> 0 ω_1	> 0 ω_2	$\rightarrow \omega_1$

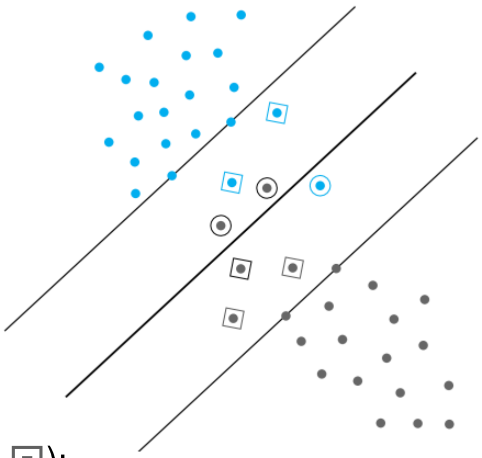
- Sort of a voting scheme
- Training and inference can be slow for N, M large

$g_{12}(x)$	> 0	< 0	
$g_{13}(x)$	> 0		< 0
$g_{23}(x)$		> 0	< 0
	ω_1	ω_2	ω_3

Contents

Non-linearly separable classes

- When the classes are not linearly separable, the original setup is **no longer valid**
  - Any attempt to draw a hyperplane will never end up with a class separation band
    $$w^T x + w_0 = \pm 1$$
    with no data points inside it



  - For this case, we have the following classes of samples:
    1) Points that fall outside the band, at the correct side (•, •):
       $$y_i(w^T x_i + w_0) \geq 1$$
    2) Points that fall inside the band, also at the correct side (▣, ▣):
       $$0 \leq y_i(w^T x_i + w_0) < 1$$
    3) Points that are missclassified (⊙, ⊙):
       $$y_i(w^T x_i + w_0) < 0$$
  - This can be summarized by introducing a new set of variables ξ_i (**slack variables**) such that $y_i(w^T x_i + w_0) \geq 1 - \xi_i$
    - In this way:
      $$\begin{array}{ll} (1) & \xi_i = 0 \\ (2) & 0 < \xi_i \leq 1 \\ (3) & \xi_i > 1 \end{array}$$

Non-linearly separable classes

- The **goal** is now
  - to make the margin as large as possible, but at the same time
  - to keep the number of samples with $\xi_i > 0$ as small as possible

$$\begin{aligned} \min J(w, w_0, \xi) &= \frac{1}{2} w^T w + C \textstyle\sum_{i=1}^N \xi_i \\ \text{subject to} \quad & y_i(w^T x_i + w_0) \geq 1 - \xi_i, \ i = 1, \ldots, N \\ & \xi_i \geq 0, \ i = 1, \ldots, N \end{aligned}$$

SOFT MARGIN problem
versus
HARD MARGIN problem

where C is a positive constant that controls the relative influence of the ξ term

- The problem is solved by a **Lagrangian** and the **Karush-Kuhn-Tucker conditions**:

$$L(w, w_0, \xi, \lambda, \mu) = \frac{1}{2} w^T w + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \lambda_i \left[y_i(w^T x_i + w_0) - 1 + \xi_i \right] - \sum_{i=1}^N \mu_i \xi_i$$

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \textstyle\sum_{i=1}^N \lambda_i y_i x_i$$

$$\frac{\partial L}{\partial w_0} = 0 \Rightarrow \textstyle\sum_{i=1}^N \lambda_i y_i = 0$$

$$\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow C - \mu_i - \lambda_i = 0, \ i = 1, \ldots, N$$

$$\lambda_i \left[y_i(w^T x_i + w_0) - 1 + \xi_i \right] = 0, \ i = 1, \ldots, N$$

$$\mu_i \xi_i = 0, \ i = 1, \ldots, N$$

$$\lambda_i \geq 0, \ \mu_i \geq 0, \ i = 1, \ldots, N$$

Non-linearly separable classes

- The corresponding **Wolfe dual representation** is obtained from the primal problem:

$$\min\ L(w, w_0, \xi, \lambda, \mu) = \frac{1}{2}w^T w + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\lambda_i\left[y_i(w^T x_i + w_0) - 1 + \xi_i\right] - \sum_{i=1}^{N}\mu_i\xi_i$$

subject to $w = \sum_{i=1}^{N}\lambda_i y_i x_i$

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

$C - \mu_i - \lambda_i = 0,\ i = 1, \ldots, N$

$\lambda_i \geq 0,\ \mu_i \geq 0,\ i = 1, \ldots, N$

… substituting the above equality constraints into the Lagrangian to end up with:

$$\max\ L(\lambda) = \sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y_i y_j x_i^T x_j$$

$$\sum_{i=1}^{N}\lambda_i y_i = 0 \text{ and } 0 \leq \lambda_i \leq C,\ i = 1, \ldots, N$$

– The only difference with the linearly-separable case is the bound C on λ_i.

Non-linearly separable classes

- Summing up:

Hard margin formulation

$$\min \quad J(w, w_0) = \frac{1}{2} w^T w$$

$$\text{s.t.} \quad y_i(w^T x_i + w_0) \geq 1, \;\; i = 1, \ldots, N$$

$\Downarrow$ *Wolfe dual representation*

$$\max \; L(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j$$

$$\text{s.t.} \;\; \textstyle\sum_{i=1}^{N} \lambda_i y_i = 0$$
$$\lambda_i \geq 0, \;\; i = 1, \ldots, N$$

Soft margin formulation

$$\min \; J(w, \xi) = \frac{1}{2} w^T w + C \textstyle\sum_{i=1}^{N} \xi_i$$

$$\text{s.t.} \; y_i(w^T x_i + w_0) \geq 1 - \xi_i, \;\; i = 1, \ldots, N$$
$$\xi_i \geq 0, \;\; i = 1, \ldots, N$$

$\Downarrow$ *Wolfe dual representation*

$$\max \; L(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j$$

$$\text{s.t.} \;\; \textstyle\sum_{i=1}^{N} \lambda_i y_i = 0$$
$$0 \leq \lambda_i \leq C, \;\; i = 1, \ldots, N$$

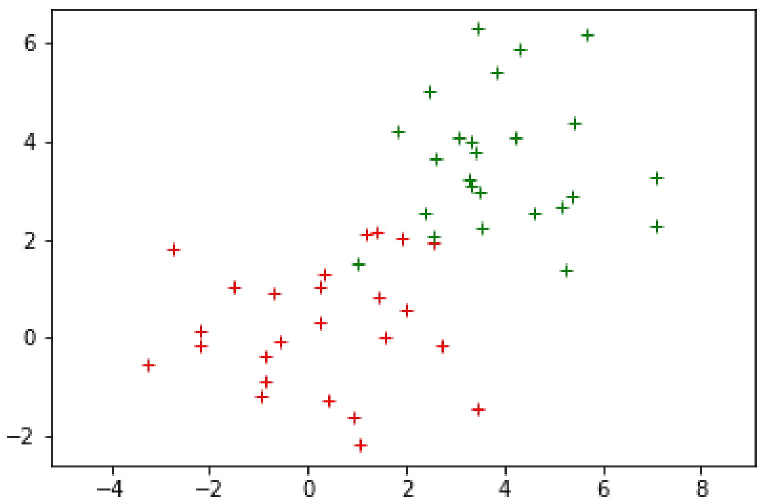
$$(1) \; \text{solve for } \lambda$$

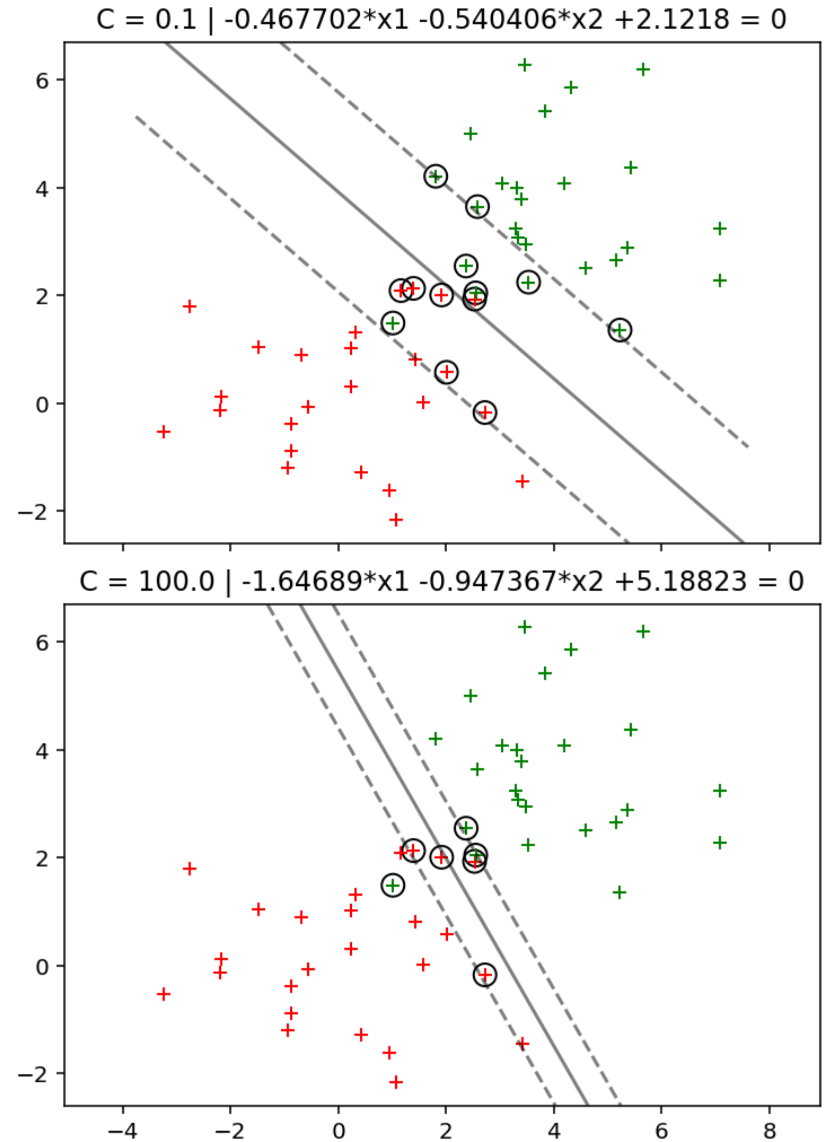
$$(2) \; w = \sum_{i|\lambda_i \neq 0} \lambda_i y_i x_i$$

$$(3) \; w_0 = \frac{1}{N_\lambda} \sum_{i|\lambda_i \neq 0} y_i - \sum_{i,j|\lambda_i,\lambda_j \neq 0} \lambda_j y_j x_j^T x_i$$

$$\underline{\text{classify}}\text{:} \;\; \text{sign}(w^T x + w_0) \equiv \text{sign}\left(\sum_{i|\lambda_i \neq 0} \lambda_i y_i x_i^T x + w_0 \right)$$

- <u>Example 2</u>: derive the SVM corresponding to the next non-linearly separable classif. problem





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

$$\text{subject to} \qquad y_i(w^T x_i + w_0) \geq 1 - \xi_i, \;\; i = 1, \ldots, N$$

$$\xi_i \geq 0, \;\; i = 1, \ldots, N$$

Wolfe dual representation

$$\max \;\; L(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j$$

$$\sum_{i=1}^{N} \lambda_i y_i = 0 \text{ and } 0 \leq \lambda_i \leq C, \;\; i = 1, \ldots, N$$

Numerical Examples

- <u>Example 2</u>:

Wolfe dual representation

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

subject to $\quad y_i(w^T x_i + w_0) \geq 1 - \xi_i,\ i = 1, \ldots, N$

$\quad \xi_i \geq 0,\ i = 1, \ldots, N$

$$\max \ L(\lambda) = \sum_{i=1}^{N} \lambda_i - \tfrac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j$$

$$\sum_{i=1}^{N} \lambda_i y_i = 0 \text{ and } 0 \leq \lambda_i \leq C,\ i = 1, \ldots, N$$

using a QP solver, e.g. **cvxpy**:

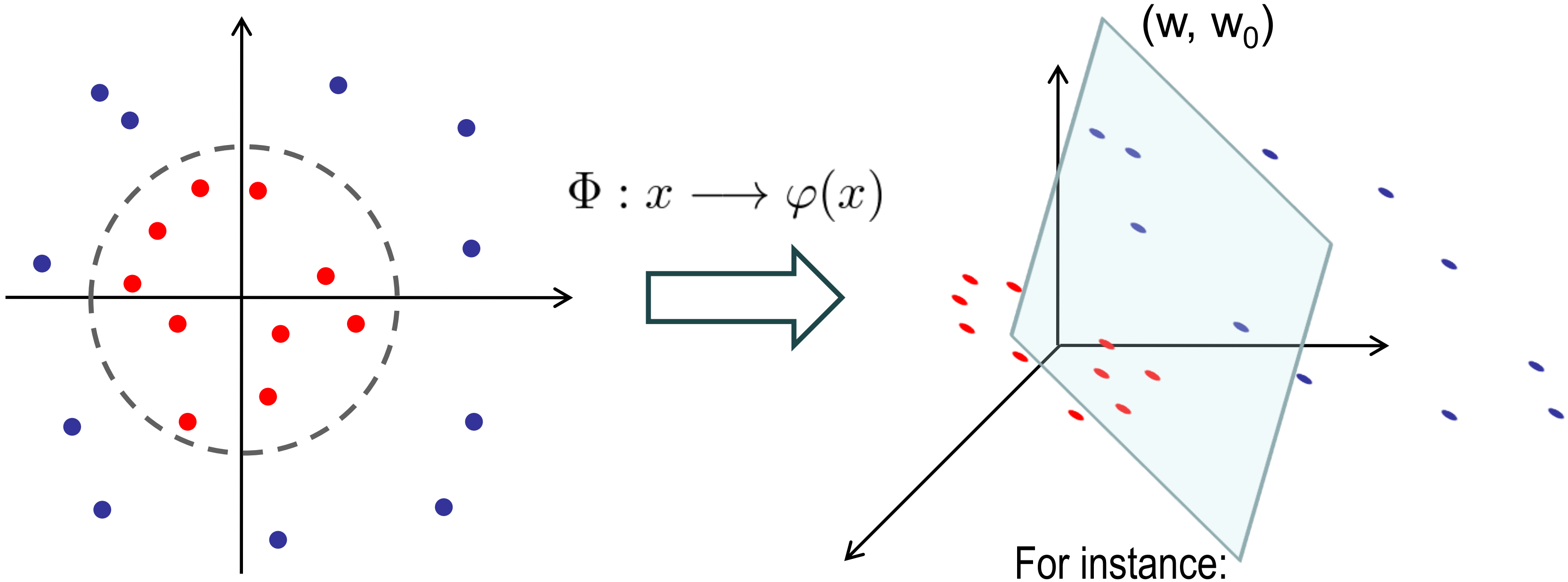
```
X = np.loadtxt('svm_samples.txt')
N = X.shape[0]
y = np.loadtxt('svm_labels.txt')
P = build_H(X, y)
A = y.reshape((1,N))
lb = np.zeros((N,1))
ub = C * np.ones((N,1))
z = cp.Variable((N,1))
P = P + (1e-8) * np.identity(N)
prob = cp.Problem(
       cp.Maximize(cp.sum(z) -
                   0.5*cp.quad_form(z,P)),
           [z >= lb, z <= ub, A@z == 0.0])
prob.solve(verbose=True, solver='SCS')
lm = z.value
ilm = (lm > 1e-4).flatten() # indices SV
```

using **scikit-learn**:

```
clf = svm.SVC(C = C, kernel = 'linear')
clf.fit(X, y)
```

Contents

- Non-linear classification problems can often be solved by **mapping the input feature space onto a larger dimensional space**, where the classes can be satisfactorily separated by a hyperplane:



For instance:

$$\begin{array}{rccc} \Phi : & \mathcal{R}^2 & \longrightarrow & \mathcal{R}^3 \\ & (x_1, x_2) & \longrightarrow & \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix} \end{array}$$

- Thanks to the SVM formulation, the cost of working in a higher dimension is not excessive, but controlled
  - This is known as the “kernel trick”

- The mapping into a higher space is incorporated in the following way:

Hard margin formulation

$$\min \quad J(w) = \frac{1}{2} w^T w$$

$$\text{s.t.} \quad y_i (w^T \Phi(x_i) + w_0) \geq 1, i = 1, \ldots, N$$

Wolfe dual representation

$$\max \; L(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \Phi(x_i)^T \Phi(x_j)$$

$$\text{s.t.} \quad \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$\lambda_i \geq 0, i = 1, \ldots, N$$

Soft margin formulation

$$\min \; J(w, \xi) = \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i$$

$$\text{s.t.} \; y_i (w^T \Phi(x_i) + w_0) \geq 1 - \xi_i, \; i = 1, \ldots, N$$

$$\xi_i \geq 0, \; i = 1, \ldots, N$$

Wolfe dual representation

$$\max \; L(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \Phi(x_i)^T \Phi(x_j)$$

$$\text{s.t.} \quad \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$0 \leq \lambda_i \leq C, \; i = 1, \ldots, N$$

(1) solve for λ

$$(2) \; w = \sum_{i|\lambda_i \neq 0} \lambda_i y_i \Phi(x_i)$$

$$(3) \; w_0 = \frac{1}{N_\lambda} \sum_{i|\lambda_i \neq 0} y_i - \sum_{i,j|\lambda_i, \lambda_j \neq 0} \lambda_j y_j \Phi(x_j)^T \Phi(x_i)$$

classify: $\text{sign}(w^T \Phi(x) + w_0) \equiv$

$$\text{sign} \left(\sum_{i|\lambda_i \neq 0} \lambda_i y_i \Phi(x_i)^T \Phi(x) + w_0 \right)$$

- For instance:

$$\Phi : \quad \mathcal{R}^2 \longrightarrow \mathcal{R}^3$$
$$(x_1, x_2) \longrightarrow \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$

$$\begin{aligned}\Phi(x)^T\Phi(z) &= \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)\begin{pmatrix} z_1^2 \\ \sqrt{2}z_1z_2 \\ z_2^2 \end{pmatrix} \\ &= x_1^2z_1^2 + 2x_1x_2z_1z_2 + x_2^2z_2^2 \\ &= (x_1z_1 + x_2z_2)^2 = \left(x^Tz\right)^2 = K_{hp}(x,z)\end{aligned}$$

<u>Kernel trick</u>: one can operate in the original space (less computation) instead of operating in the larger-dimensional space, but with the advantages of the latter

- This and other functions known as **kernels** satisfy the following condition:

$$K(x,z) = \Phi(x)^T\Phi(z)$$

(Mercer's theorem characterizes these functions)

- This is the case of:

Linear kernel	$K_{ln}(x,z) = x^Tz$
(homogeneous) Polynomial kernel	$K_{hp}(x,z) = (x^Tz)^q, \; q > 0$
(inhomogeneous) Polynomial kernel	$K_{ip}(x,z) = (\gamma x^Tz + r)^q, \; q > 0, \; r$ usually 1
(Gaussian) Radial Basis Function kernel	$K_{rbf}(x,z) = e^{-\frac{\Vert x-z\Vert^2}{2\sigma^2}} = e^{-\gamma\Vert x-z\Vert^2}$

[in this last case, the higher-dimensional feature space $\Phi(x)$ is infinite dimensional]

and others …

Non-linear SVM

- Another example: $\Phi(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)^T$

$$\Phi(x)^T\Phi(z) = (1 + x^T z)^2 = K_{ih}(x, z)$$

- In general, the **expansion** of an L-variate M-degree inhomogeneous polynomial is: (in the following, all coefficients are assumed 1 for simplicity)

$$\Pi^M(x_1, \ldots, x_L) = 1 + \sum_{i=1}^{L} x_i + \sum_{\substack{i,j=1 \\ a+b=2 \\ a\geq 0, b\geq 0}}^{L} x_i^a x_j^b + \ldots + \sum_{\substack{i,j,\cdots=1 \\ a+b+\cdots=M \\ a\geq 0, b\geq 0, \ldots}}^{L} x_i^a x_j^b \ldots$$

- The **number of terms** of Π^M(x), and Φ(x), is thus:

$$1 + \sum_{i=1}^{M} \mathrm{CR}_{L,i} = 1 + \mathrm{CR}_{L,1} + \mathrm{CR}_{L,2} + \cdots + \mathrm{CR}_{L,M} = \sum_{i=0}^{M} \binom{L+i-1}{i}$$

$$= \binom{L-1}{0} + \binom{L}{1} + \binom{L+1}{2} + \cdots + \binom{L+M-1}{M} = \frac{(L+M)!}{M!L!}$$

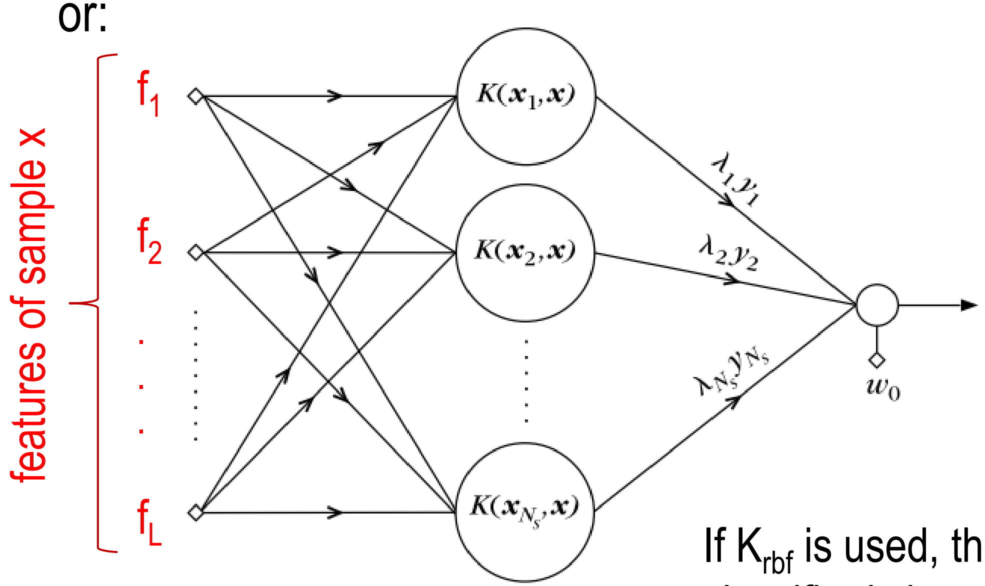
- For instance, for **L = 10** and **M = 4**, Φ(x) dimension becomes 1001:
  - computing $\Phi(x)^T\Phi(z)$ means a dot product involving 1001-component vectors,
  - while $(1 + x^Tz)^4$ represents a dot product involving 10-component vectors

Non-linear SVM

- Apart from the benefits of working in a higher number of dimensions at almost no cost, with the “kernel trick” the **classification** operation becomes:

$$\operatorname{sign}\left(\overbrace{\sum_{i|\lambda_i \neq 0} \lambda_i y_i K(x_i, x)}^{w^T\Phi(x)} + w_0\right) > 0\ (< 0) \Rightarrow x \rightarrow \omega_1(\omega_2)$$

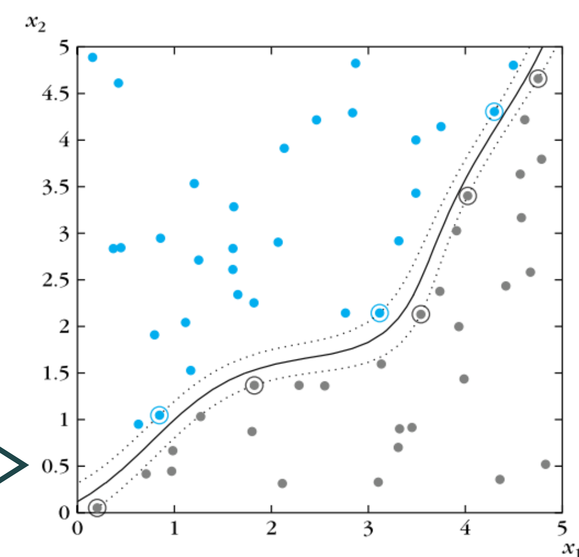
or:



$x = (f_1, f_2, \ldots, f_L)$

N_s: num. support vectors

If K_{rbf} is used, the classifier is known as an **RBF network**



Contents

- Formulation of the SVM problem for linearly separable classes
- SVM training for linearly separable classes
- Non-linearly separable classes
- Non-linear SVM
- Numerical examples
- Final remarks

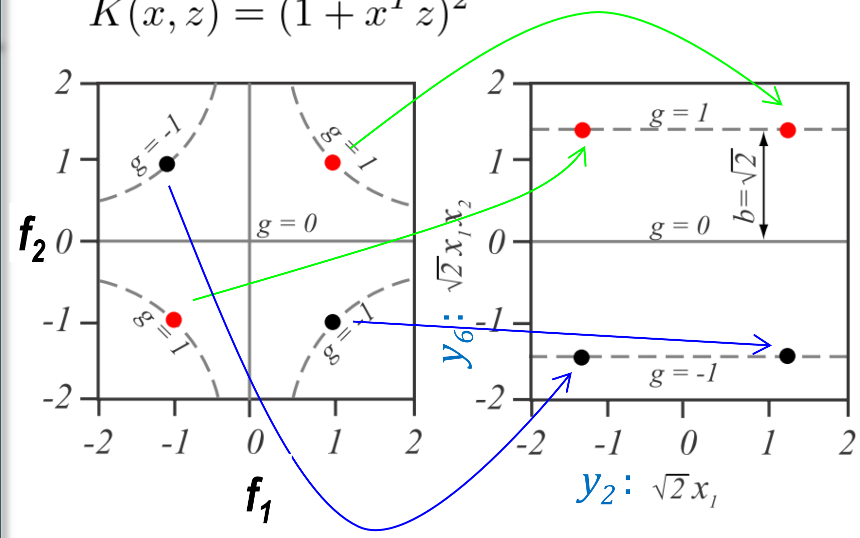
- Example 3: derive the SVM corresponding to the next 2-class classification problem

$\omega_1 = \{ (1,1)^T, (-1,-1)^T \}$ (●)

$\omega_2 = \{ (1,-1)^T, (-1,1)^T \}$ (●)

$$\Phi(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$K(x,z) = (1 + x^T z)^2$$



Wolfe dual representation

$$\min \quad J(w, w_0) = \frac{1}{2} w^T w$$

$$\text{s.t.} \quad y_i(w^T \Phi(x_i) + w_0) \geq 1, \ i = 1, \dots, N$$

$$\max \ L(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j K(x_i, x_j)$$

$$\text{s.t.} \ \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$\lambda_i \geq 0, \ i = 1, \dots, N$$

$$(1) \ H = \begin{bmatrix} y_1 y_1 K(x_1, x_1) & y_1 y_2 K(x_1, x_2) & \dots & y_1 y_N K(x_1, x_N) \\ y_2 y_1 K(x_2, x_1) & y_2 y_2 K(x_2, x_2) & \dots & y_2 y_N K(x_2, x_N) \\ \vdots & \vdots & \ddots & \vdots \\ y_N y_1 K(x_N, x_1) & y_N y_2 K(x_N, x_2) & \dots & y_N y_N K(x_N, x_N) \end{bmatrix}$$

$$(2) \ w = \sum_{i|\lambda_i \neq 0} \lambda_i y_i \Phi(x_i)$$

$$(3) \ w_0 = \frac{1}{N_\lambda} \sum_{i|\lambda_i \neq 0} (y_i - \sum_{j|\lambda_j \neq 0} \lambda_j y_j K(x_j, x_i))$$

- **Solution**: $\omega_1 : x_1 = (1,1)^T, x_3 = (-1,-1)^T \Rightarrow y_1 = +1, y_3 = +1$
  $\omega_2 : x_2 = (1,-1)^T, x_4 = (-1,1)^T \Rightarrow y_2 = -1, y_4 = -1$

$$K(x,z) = (1 + x^T z)^2$$

$$H = \begin{bmatrix} y_1 y_1 K(x_1,x_1) & y_1 y_2 K(x_1,x_2) & y_1 y_3 K(x_1,x_3) & y_1 y_4 K(x_1,x_4) \\ y_2 y_1 K(x_2,x_1) & y_2 y_2 K(x_2,x_2) & y_2 y_3 K(x_2,x_3) & y_2 y_4 K(x_2,x_4) \\ y_3 y_1 K(x_4,x_1) & y_3 y_2 K(x_3,x_2) & y_3 y_4 K(x_3,x_3) & y_3 y_4 K(x_3,x_4) \\ y_4 y_1 K(x_4,x_1) & y_4 y_2 K(x_4,x_2) & y_4 y_4 K(x_4,x_3) & y_4 y_4 K(x_4,x_4) \end{bmatrix} = \begin{bmatrix} 9 & -1 & 1 & -1 \\ -1 & 9 & -1 & 1 \\ 1 & -1 & 9 & -1 \\ -1 & 1 & -1 & 9 \end{bmatrix}$$

using a QP solver, e.g. **cvxpy** :

$$\min \frac{1}{2} z^T P z + q^T z$$
$$\text{s.t.} \quad z \geq 0$$
$$Az = b$$

```
P = build_H_wpk(X, y, g=1, r=1, q=2)
A = y.reshape((1,4))
z = cp.Variable((4,1))
prob = cp.Problem(cp.Minimize(0.5 * cp.quad_form(z,P) - cp.sum(z)),
                  [z >= 0, A @ z == 0])
prob.solve()
l = z.value
ilm = (lm > 1e-6).flatten()
```

$$K_{ip}(x,z) = (\gamma x^T z + r)^q$$

$$\Phi(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1 x_2)$$

$$K(x,z) = (1 + x^T z)^2$$

$$\Rightarrow \begin{cases} \lambda_1 = 0.125 \\ \lambda_2 = 0.125 \\ \lambda_3 = 0.125 \\ \lambda_4 = 0.125 \end{cases}$$

$$w = \sum_{i|\lambda_i \neq 0} \lambda_i y_i \Phi(x_i) = \lambda_1 \begin{bmatrix} 1 \\ \sqrt{2} \\ \sqrt{2} \\ 1 \\ 1 \\ \sqrt{2} \end{bmatrix} - \lambda_2 \begin{bmatrix} 1 \\ \sqrt{2} \\ -\sqrt{2} \\ 1 \\ 1 \\ -\sqrt{2} \end{bmatrix} + \lambda_3 \begin{bmatrix} 1 \\ -\sqrt{2} \\ -\sqrt{2} \\ 1 \\ 1 \\ \sqrt{2} \end{bmatrix} - \lambda_4 \begin{bmatrix} 1 \\ -\sqrt{2} \\ \sqrt{2} \\ 1 \\ 1 \\ -\sqrt{2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

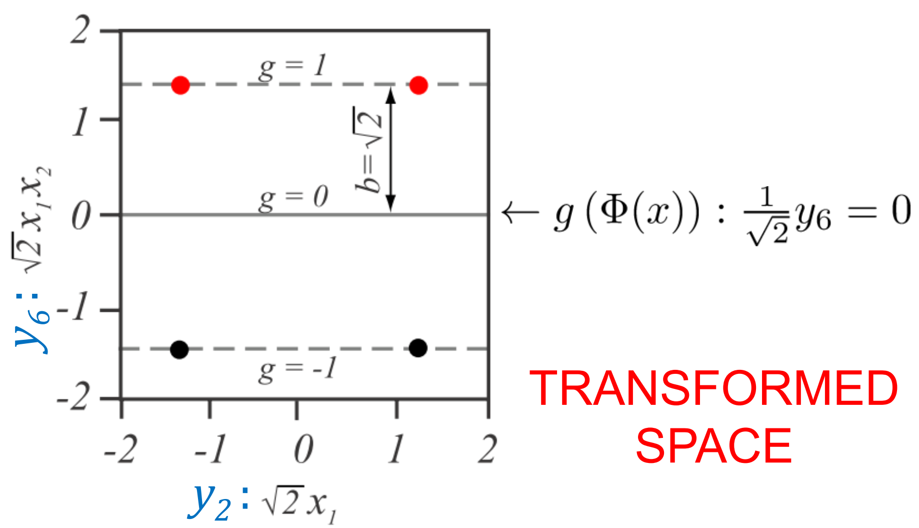
$$w_0 = \underbrace{1}_{y_1} - \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}^T \begin{bmatrix} 1 \\ \sqrt{2} \\ \sqrt{2} \\ 1 \\ 1 \\ \sqrt{2} \end{bmatrix}}_{w^T \Phi(x_1)} = 0$$

- **Solution**:

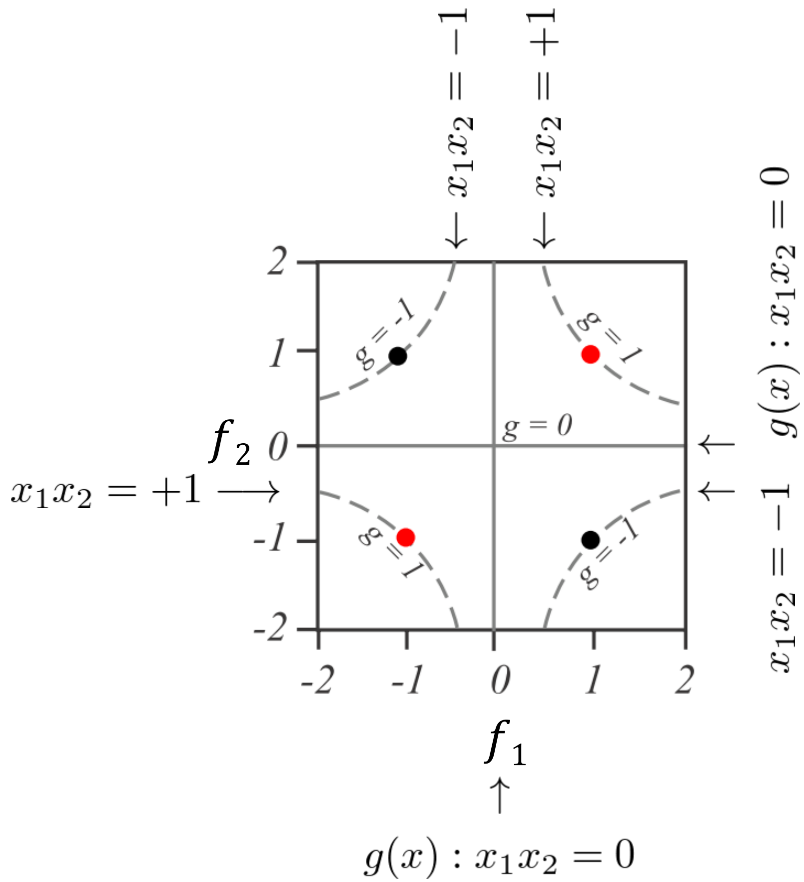
$$\Phi(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$
$$w = (0,0,0,0,0,\frac{1}{\sqrt{2}})$$
$$\Rightarrow \text{SVM}: \frac{1}{\sqrt{2}}\left(\sqrt{2}x_1x_2\right) = x_1x_2 = 0$$

and the SV lie on $x_1x_2 = \pm 1$

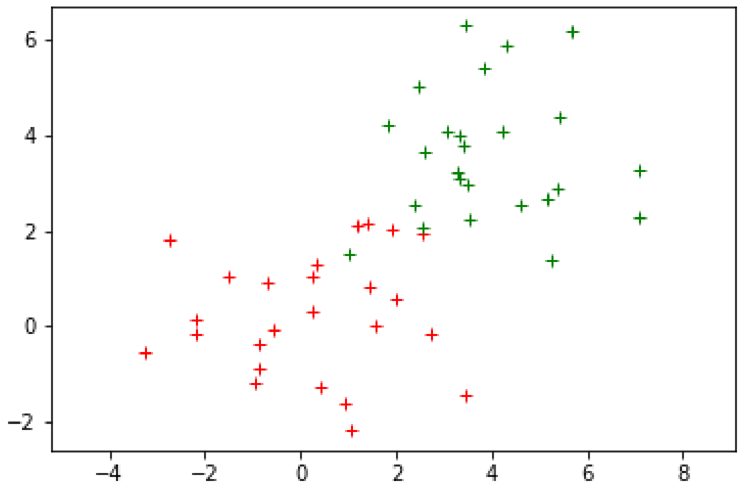
ORIGINAL SPACE

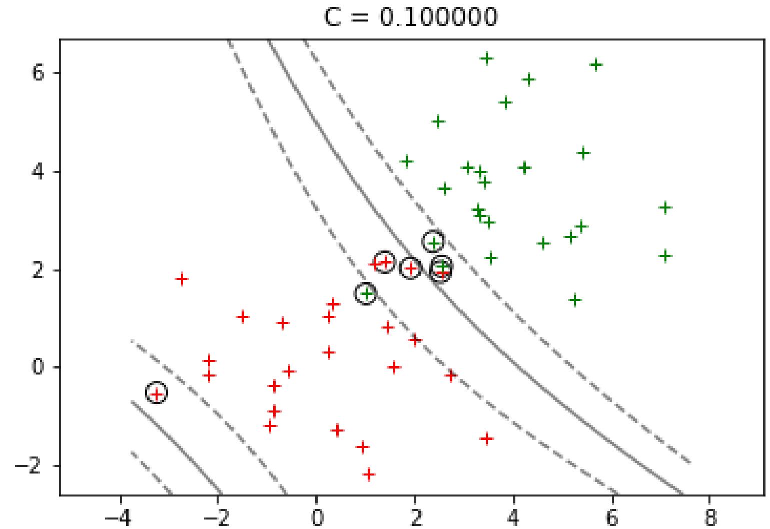


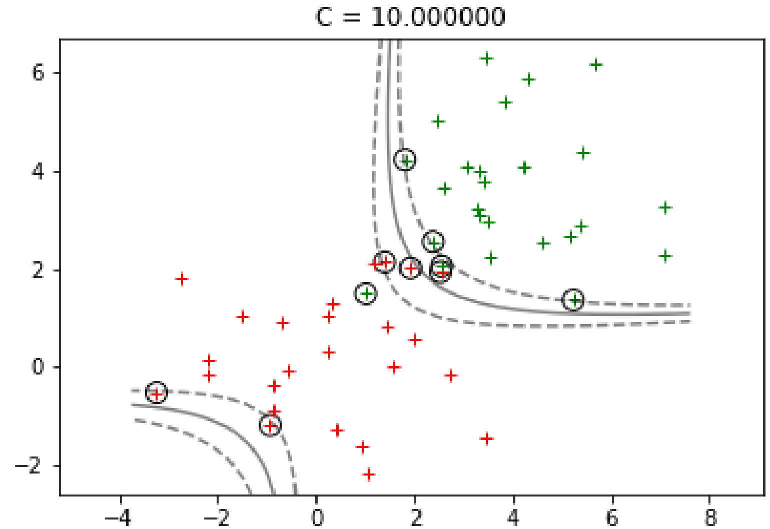
TRANSFORMED SPACE



- Example 4: derive the SVM for the following classif. problem using $K(x,z) = (1 + \ x^T z)^2$







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

$$\text{subject to} \quad y_i(w^T \Phi(x_i) + w_0) \geq 1 - \xi_i, \ i = 1, \dots, N$$

$$\xi_i \geq 0, \ i = 1, \dots, N$$

Wolfe dual representation

$$\max \ L(\lambda) = \sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y_i y_j K(x_i, x_j)$$

$$\sum_{i=1}^N \lambda_i y_i = 0 \text{ and } 0 \leq \lambda_i \leq C, \ i = 1, \dots, N$$

- Example 4:

Wolfe dual representation

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

$$\text{subject to} \quad y_i(w^T \Phi(x_i) + w_0) \geq 1 - \xi_i,\ i = 1, \ldots, N$$

$$\xi_i \geq 0,\ i = 1, \ldots, N$$

$$\max\ L(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j K(x_i, x_j)$$

$$\sum_{i=1}^{N} \lambda_i y_i = 0 \text{ and } 0 \leq \lambda_i \leq C,\ i = 1, \ldots, N$$

using a QP solver, e.g. **cvxpy**:

```
X = np.loadtxt('svm_samples.txt')
y = np.loadtxt('svm_labels.txt')
P = build_H_wpk(X, y, g=1, r=1, q=2)
A = y.reshape((1,N))
lb = np.zeros((N,));
ub = C * np.ones((N,))
z = cp.Variable((N,1))
P = P + (1e-6) * np.identity(N)
prob = cp.Problem(
       cp.Minimize(0.5 * cp.quad_form(z, P)
                        - cp.sum(z)),
       [z >= lb, z <= ub, A @ z = 0] )
prob.solve(solver='SCS')
ilm = (lm > 1e-6).flatten() # indices SV
```

standard formulation

$$\min \frac{1}{2} z^T P z + q^T z$$

$$\text{s.t.} \quad Gz \leq h$$

$$Az = b$$

$$lb \leq z \leq ub$$

using **scikit-learn**:

```
clf = svm.SVC(C = C, kernel = 'poly',
      degree = 2, coef0 = 1, gamma = 1)
clf.fit(X, y)
```

$$K(x, z) = (r + \gamma x^T z)^q = (1 + x^T z)^2$$

Final Remarks

- SVMs tend to be **less prone to overfitting** than other methods
  - Because the classifier resulting from the SVM approach **depends only on the SV**, which are the **most significative** patterns for the classification task
  - Besides, the margin band **contributes to the generalization performance**
  - As a consequence, in general, they **exhibit good generalization performance**
- The **complexity** of the classifier depends more on the number of SV than on the dimensionality of the feature space
  - Thanks to the SVM formulation and the kernel trick, working in a higher dimension is almost at **zero cost**
- However:
  - There is not an efficient practical method for **choosing the best kernel**
  - Besides, once a kernel has been chosen, its parameters' values, **hyperparameters**, have to be selected
    - They are crucial to the generalization capabilities of the classifier
  - As a consequence, the most common procedure is to **solve the SVM task for different sets of parameters** (grid search)

Instance-based learning: Support Vector Machines

Universitat de les Illes Balears
Departament de Ciències Matemàtiques i Informàtica

11752 Aprendizaje Automático
11752 Machine Learning
Máster Universitario
en Sistemas Inteligentes

Alberto ORTIZ RODRÍGUEZ