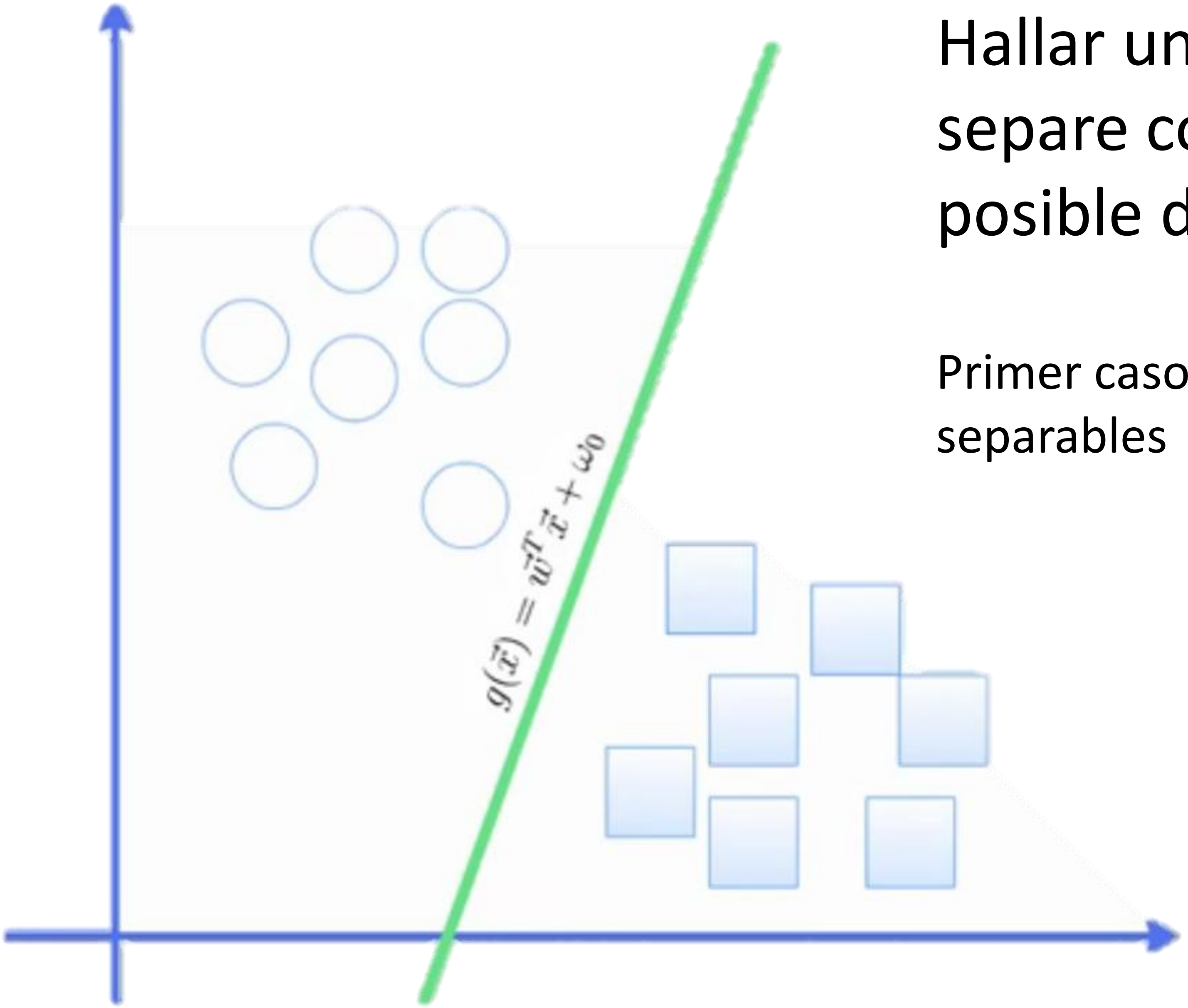
Support Vector Machine Clasificador

Alexis Carrillo

Contenido

- Large margin clasificator
- Soft Margin Clasificator
- Non linear clasificator (Kernel machines)



Hallar un hiperplano que separe con el mayor margen posible datos de 2 clases

Primer caso: clases linealmente separables

Let us start again with two classes and use labels $-1/+1$ for the two classes. The sample is $\mathcal{X} = \{\boldsymbol{x}^t, r^t\}$ where $r^t = +1$ if $\boldsymbol{x}^t \in C_1$ and $r^t = -1$ if $\boldsymbol{x}^t \in C_2$. We would like to find $\boldsymbol{w}$ and w_0 such that

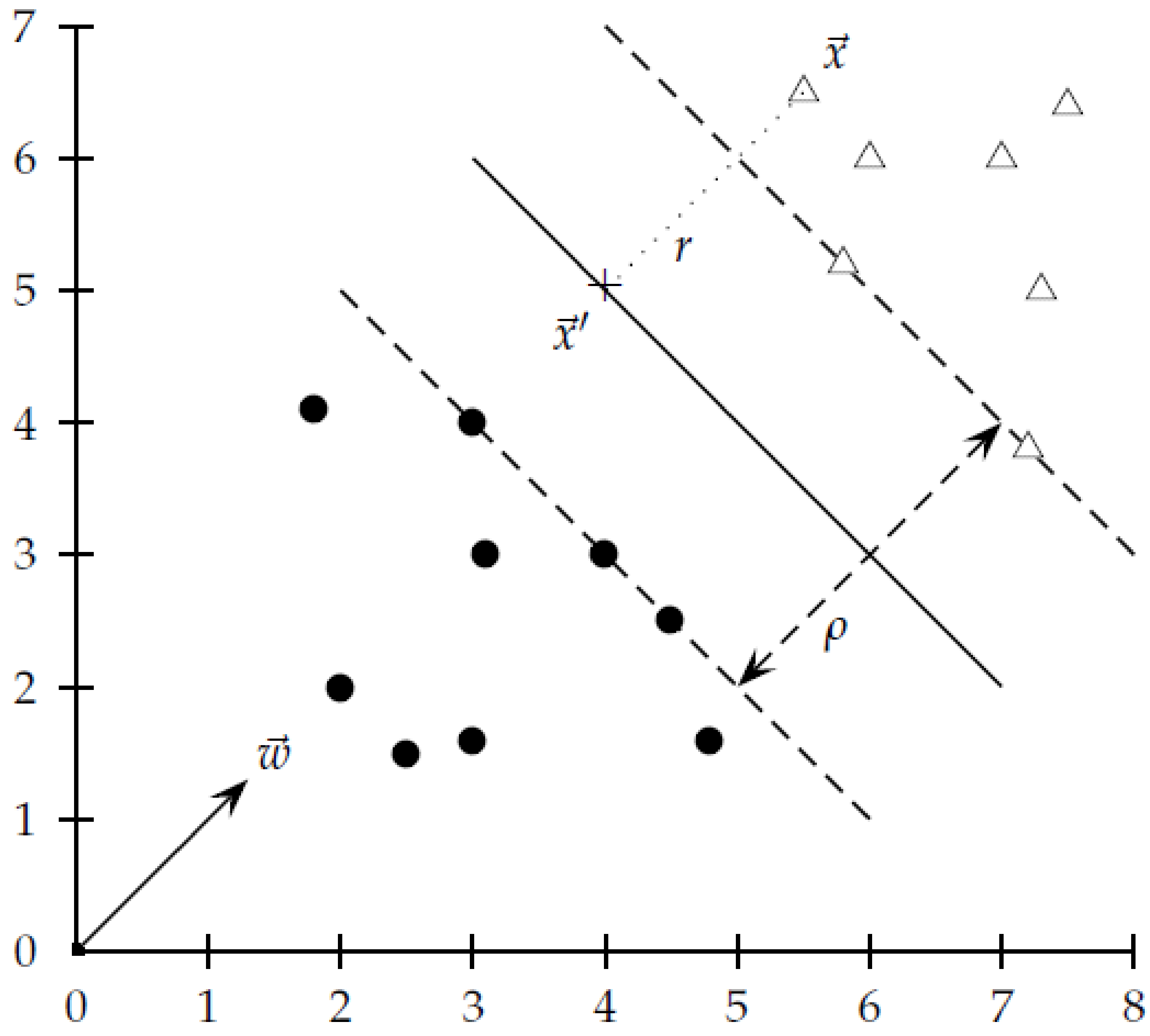
$$\begin{aligned} \boldsymbol{w}^T\boldsymbol{x}^t + w_0 &\geq +1 \quad \text{for} \quad r^t = +1 \\ \boldsymbol{w}^T\boldsymbol{x}^t + w_0 &\leq -1 \quad \text{for} \quad r^t = -1 \end{aligned}$$

which can be rewritten as

$$r^t(\boldsymbol{w}^T\boldsymbol{x}^t + w_0) \geq +1$$

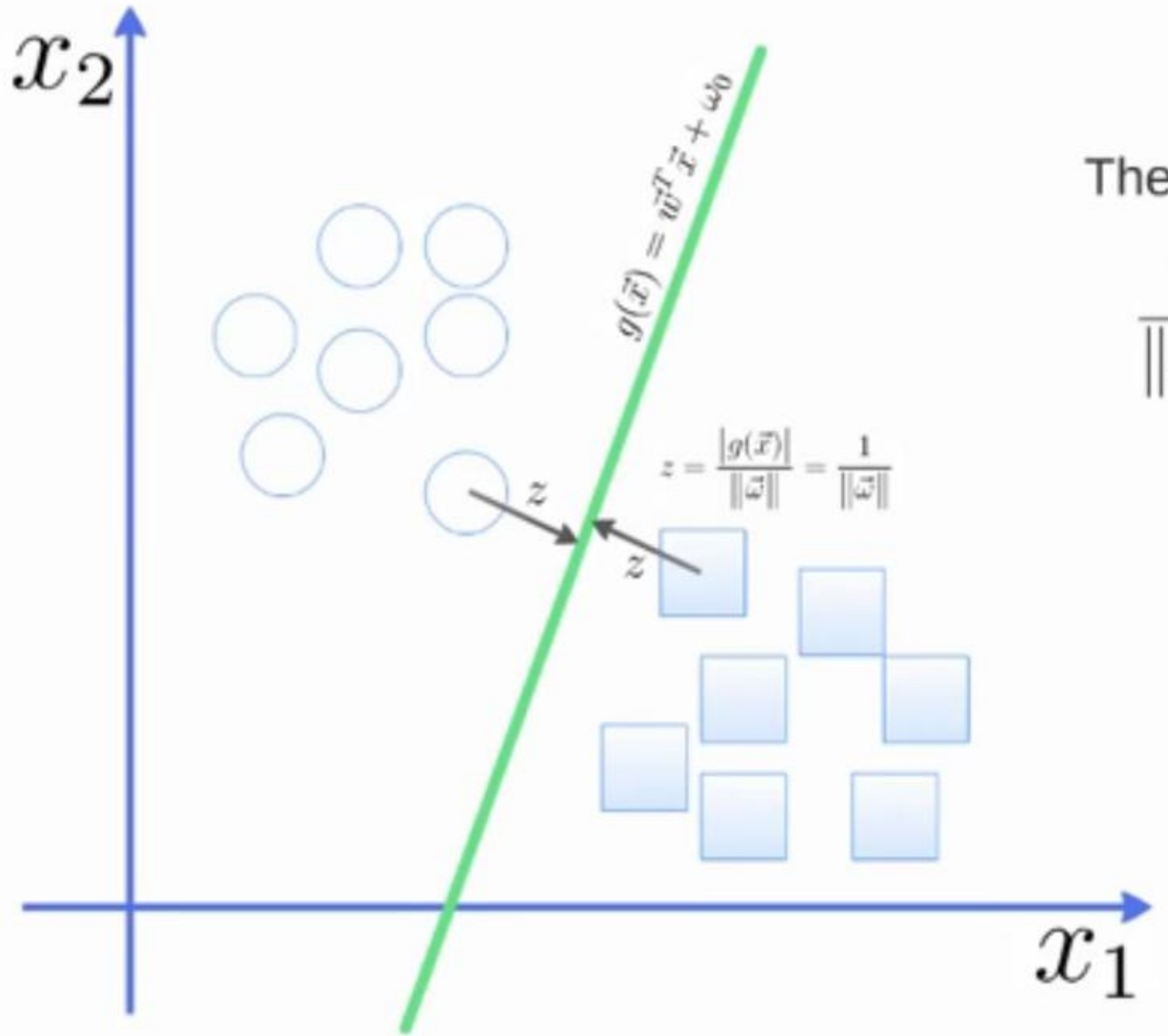
Note that we do not simply require

$$r^t(\boldsymbol{w}^T\boldsymbol{x}^t + w_0) \geq 0$$



7
6
5
4
3
2
1
0
0
1
2
3
4
5
6
7
8
$\vec{x}$
r
$\vec{x}'$
ρ
$\vec{w}$

$$g(\vec{x}) \geqslant 1, \quad \forall \vec{x} \in \text{class } 1$$
$$g(\vec{x}) \leqslant -1, \quad \forall \vec{x} \in \text{class } 2$$



The total margin is computed by

$$\frac{1}{\|\vec{\omega}\|} + \frac{1}{\|\vec{\omega}\|} = \frac{2}{\|\vec{\omega}\|}$$

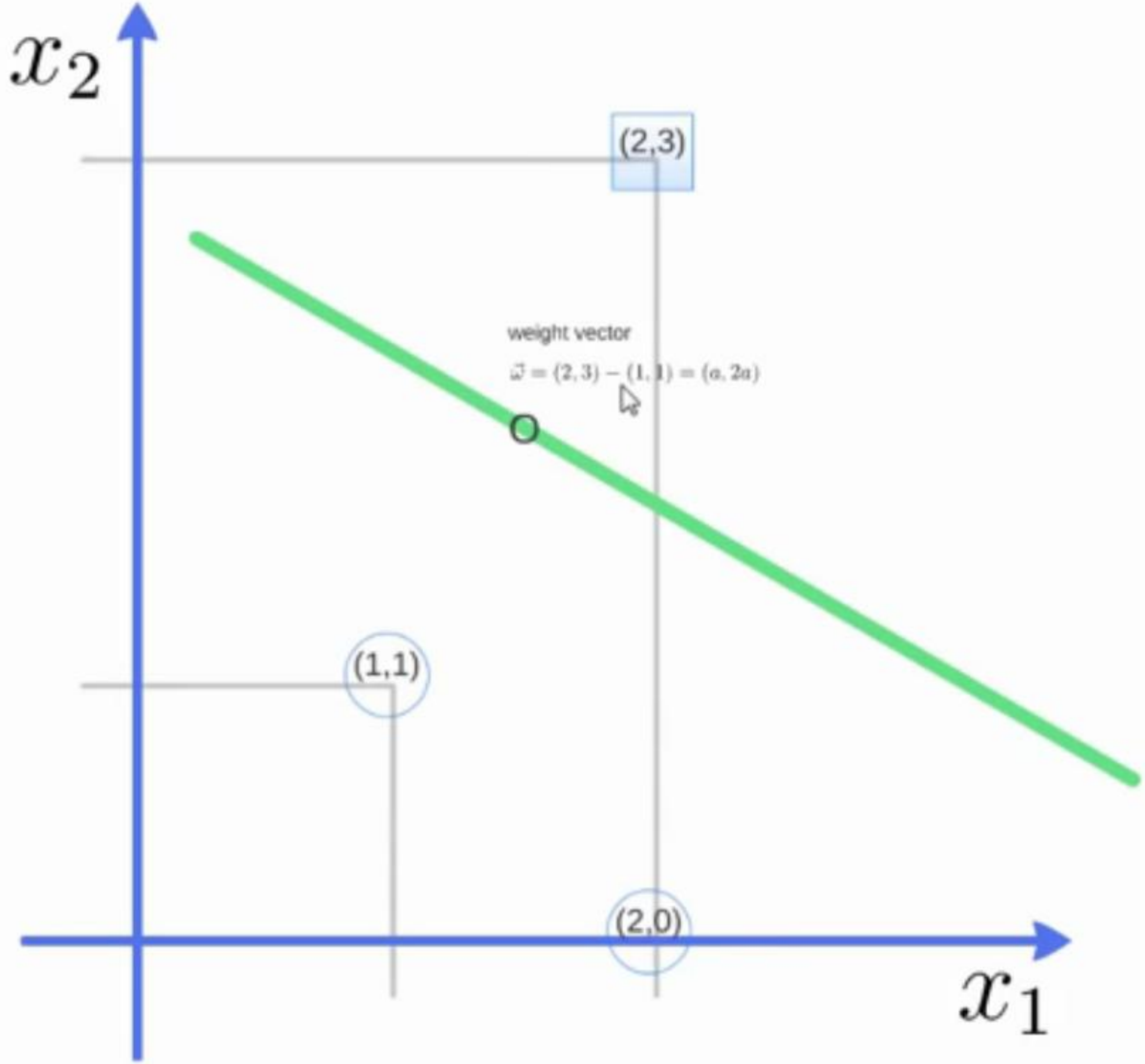
Minimizing this term will maximize the separability

Minimizing $\vec{\omega}$ is a nonlinear optimization task, solved by the Karush-Kuhn-Tucker (KKT) conditions, using Langrange multipliers λ_i

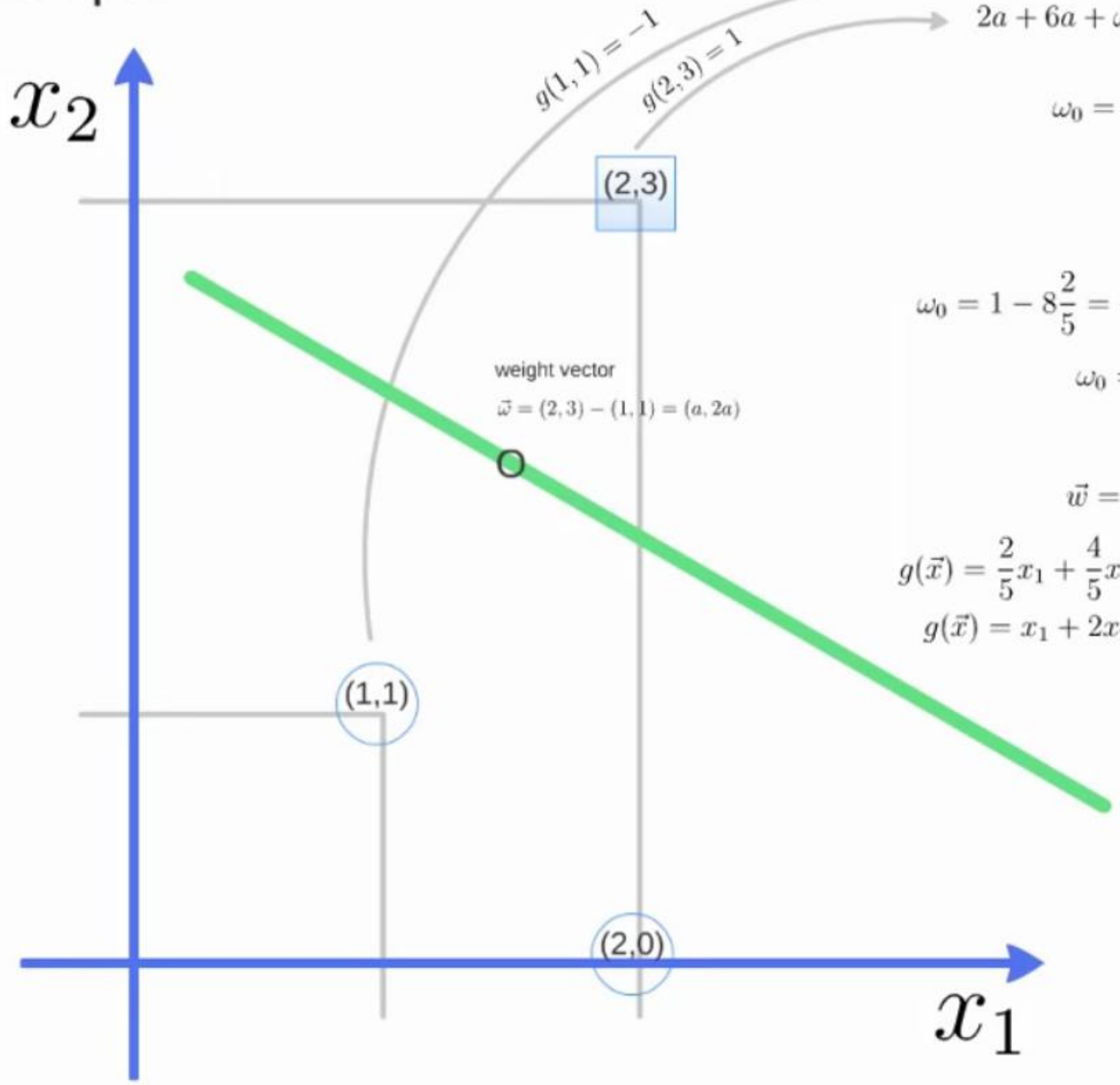
$$\vec{\omega} = \sum_{i=0}^{N} \lambda_i y_i \vec{x}_i$$

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

Example



Example



weight vector $\vec{w} = (a, 2a)$

$a + 2a + \omega_0 = -1,$ using point (1,1)

$2a + 6a + \omega_0 = 1,$ using point (2,3)

$\dots$

$\omega_0 = 1 - 8a \qquad 3a + 1 - 8a = -1$

$\vdots \qquad 5a = 2$

$$a = \frac{2}{5}$$

$$\omega_0 = 1 - 8\frac{2}{5} = \frac{5 - 16}{5}$$

$$\omega_0 = -\frac{11}{5}$$

$\dots$

$$\vec{w} = (\frac{2}{5}, \frac{4}{5})$$

$$g(\vec{x}) = \frac{2}{5}x_1 + \frac{4}{5}x_2 - \frac{11}{5}$$

$$g(\vec{x}) = x_1 + 2x_2 - 5.5$$

Soft Margin Clasificator

- Función de error para vectores que quedan dentro del margen o clasificados erróneamente.
- “Slack Variable”
- “Hinge loss”
- Equilibrio entre (mínimo) error y (máximo) margen

If the data is not linearly separable, the algorithm we discussed earlier will not work. In such a case, if the two classes are not linearly separable such that there is no hyperplane to separate them, we look for the one that incurs the least error. We define *slack variables*, $\xi^t \geq 0$, which store the deviation from the margin. There are two types of deviation: An instance may lie on the wrong side of the hyperplane and be misclassified. Or, it may be on the right side but may lie in the margin, namely, not sufficiently away from the hyperplane. Relaxing equation 13.1, we require

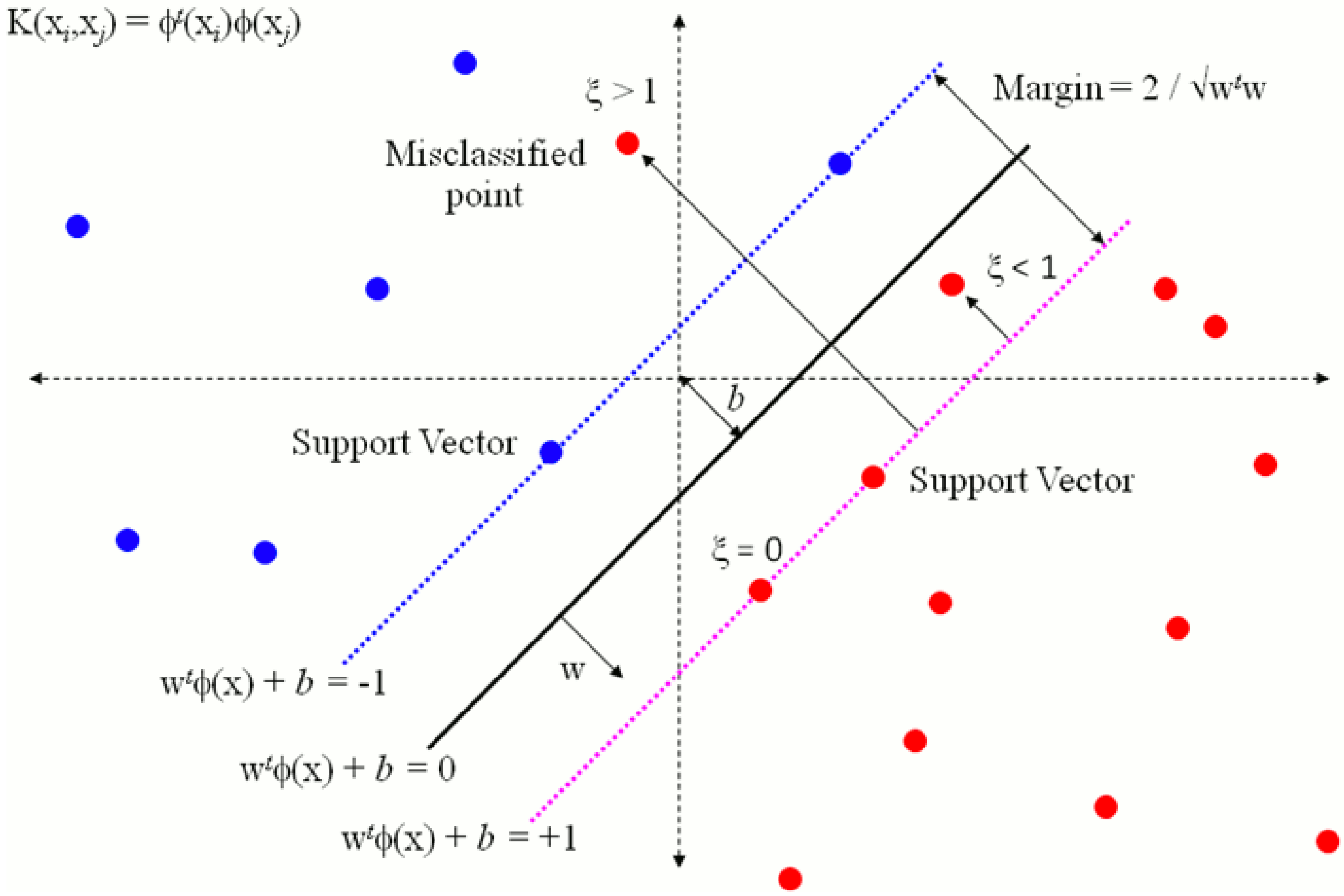
$$r^t(\mathbf{w}^T\mathbf{x}^t + w_0) \geq 1 - \xi^t$$

If $\xi^t = 0$, there is no problem with $\boldsymbol{x}^t$. If $0 < \xi^t < 1$, $\boldsymbol{x}^t$ is correctly classified but in the margin. If $\xi^t \geq 1$, $\boldsymbol{x}^t$ is misclassified (see figure 13.2). The number of misclassifications is $\#\{\xi^t > 1\}$, and the number of nonseparable points is $\#\{\xi_t > 0\}$. We define *soft error* as

$$\sum_t \xi^t$$

and add this as a penalty term:

$$L_p = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_t \xi^t$$



$K(x_i,x_j) = \phi^t(x_i)\phi(x_j)$
$\xi > 1$
Misclassified point
Margin = $2 / \sqrt{w^t w}$
$\xi < 1$
b
Support Vector
Support Vector
$\xi = 0$
w
$w^t\phi(x) + b = -1$
$w^t\phi(x) + b = 0$
$w^t\phi(x) + b = +1$

The optimization problem is then trading off how fat it can make the margin versus how many points have to be moved around to allow this margin. The margin can be less than 1 for a point $\vec{x}_i$ by setting $\xi_i > 0$, but then one pays a penalty of $C\xi_i$ in the minimization for having done that. The sum of the ξ_i gives an upper bound on the number of training errors. Soft-margin SVMs minimize training error traded off against margin. The parameter C is a *regularization* term, which provides a way to control overfitting: as C becomes large, it is unattractive to not respect the data at the cost of reducing the geometric margin; when it is small, it is easy to account for some data points with the use of slack variables and to have a fat margin placed so it models the bulk of the data.

The dual problem for soft margin classification becomes:

Find $\alpha_1, \ldots \alpha_N$ such that $\sum \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \vec{x}_i{}^{\mathrm{T}} \vec{x}_j$ is maximized, and

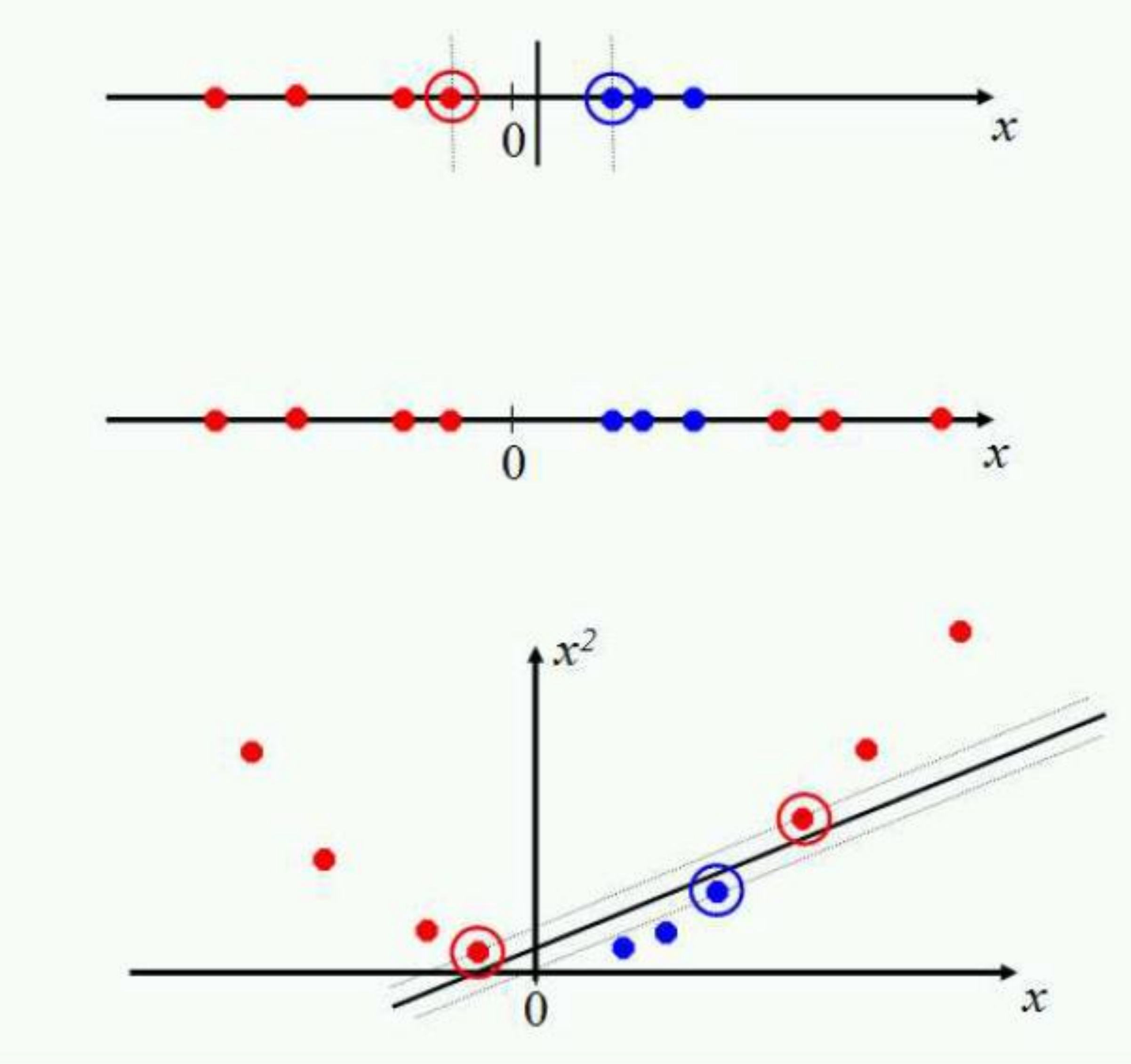
- $\sum_i \alpha_i y_i = 0$
- $0 \leq \alpha_i \leq C$ for all $1 \leq i \leq N$

Multiclass SVMs

- One Versus All slasificator (OVA).
- Multiclass SVM

Kernel

- Transformación del espacio de características con una función Φ (n+1 dimensión a partir de una medida de similitud).



0
x
0
x
x^2
0
x

$$f(\vec{x}) = \text{sign}(\sum_i \alpha_i y_i \vec{x}_i{}^{\mathrm{T}} \vec{x} + b)$$

$$f(\vec{x}) = \text{sign}(\sum_i \alpha_i y_i K(\vec{x}_i, \vec{x}) + b)$$

Now suppose we decide to map every data point into a higher dimensional space via some transformation $\Phi: \vec{x} \mapsto \phi(\vec{x})$. Then the dot product becomes $\phi(\vec{x}_i)^{\mathrm{T}}\phi(\vec{x}_j)$. If it turned out that this dot product (which is just a real number) could be computed simply and efficiently in terms of the original data points, then we wouldn't have to actually map from $\vec{x} \mapsto \phi(\vec{x})$. Rather, we could simply compute the quantity $K(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i)^{\mathrm{T}}\phi(\vec{x}_j)$, and then use the function's value in Equation (15.13). A *kernel function* K is such a function that corresponds to a dot product in some expanded feature space.

Let us say we have the new dimensions calculated through the basis functions

$\mathbf{z} = \boldsymbol{\phi}(\boldsymbol{x})$ where $z_j = \phi_j(\boldsymbol{x}), j = 1, \ldots, k$

mapping from the d-dimensional $\boldsymbol{x}$ space to the k-dimensional $\mathbf{z}$ space where we write the discriminant as

$$
\begin{aligned}
g(\mathbf{z}) &= \mathbf{w}^T \mathbf{z} \\
g(\boldsymbol{x}) &= \mathbf{w}^T \boldsymbol{\phi}(\boldsymbol{x}) \\
&= \sum_{j=1}^{k} w_j \phi_j(\boldsymbol{x})
\end{aligned}
$$

Vectorial Kernels

The most popular, general-purpose kernel functions are

- *polynomials* of degree q:

  $$K(\boldsymbol{x}^t, \boldsymbol{x}) = (\boldsymbol{x}^T \boldsymbol{x}^t + 1)^q$$

  where q is selected by the user. For example, when $q = 2$ and $d = 2$,

  $$\begin{aligned} K(\boldsymbol{x}, \boldsymbol{y}) &= (\boldsymbol{x}^T \boldsymbol{y} + 1)^2 \\ &= (x_1 y_1 + x_2 y_2 + 1)^2 \\ &= 1 + 2x_1 y_1 + 2x_2 y_2 + 2x_1 x_2 y_1 y_2 + x_1^2 y_1^2 + x_2^2 y_2^2 \end{aligned}$$

  corresponds to the inner product of the basis function (Cherkassky and Mulier 1998):

  $$\phi(\boldsymbol{x}) = [1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2]^T$$

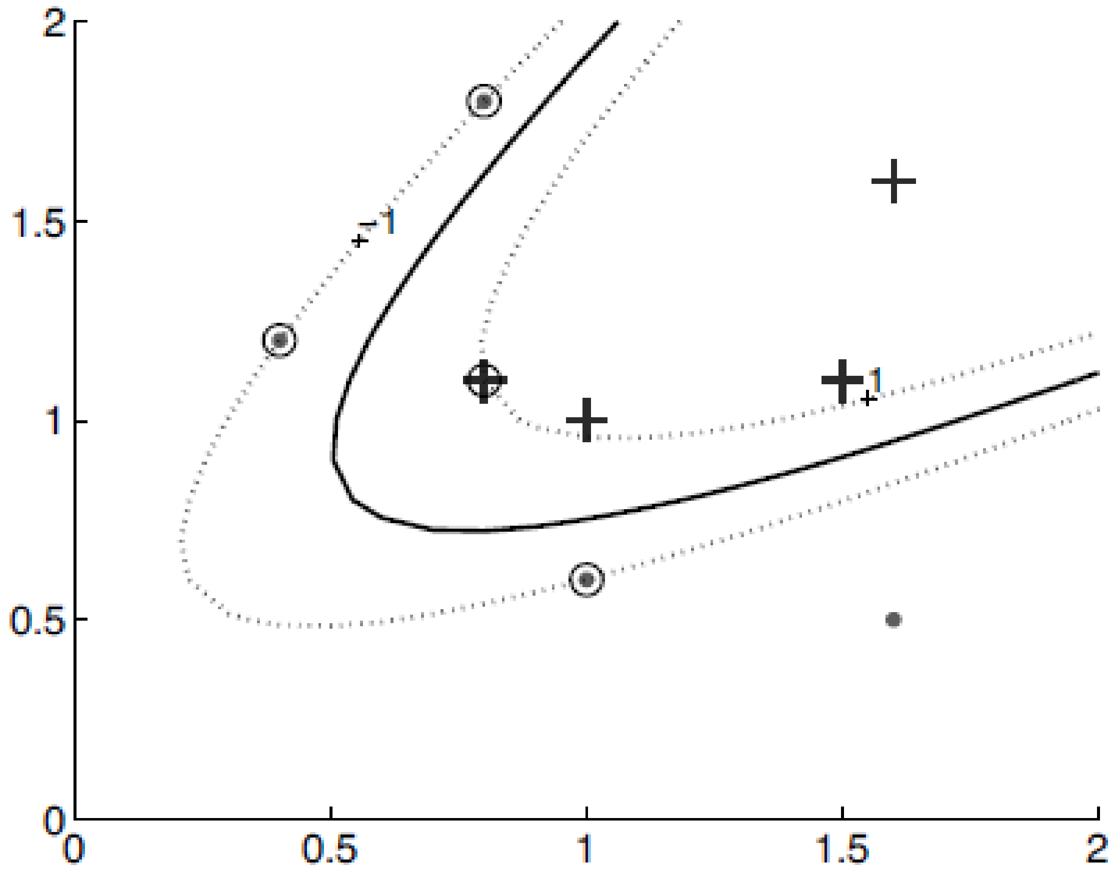


Figure 13.4 The discriminant and margins found by a polynomial kernel of degree 2. Circled instances are the support vectors.

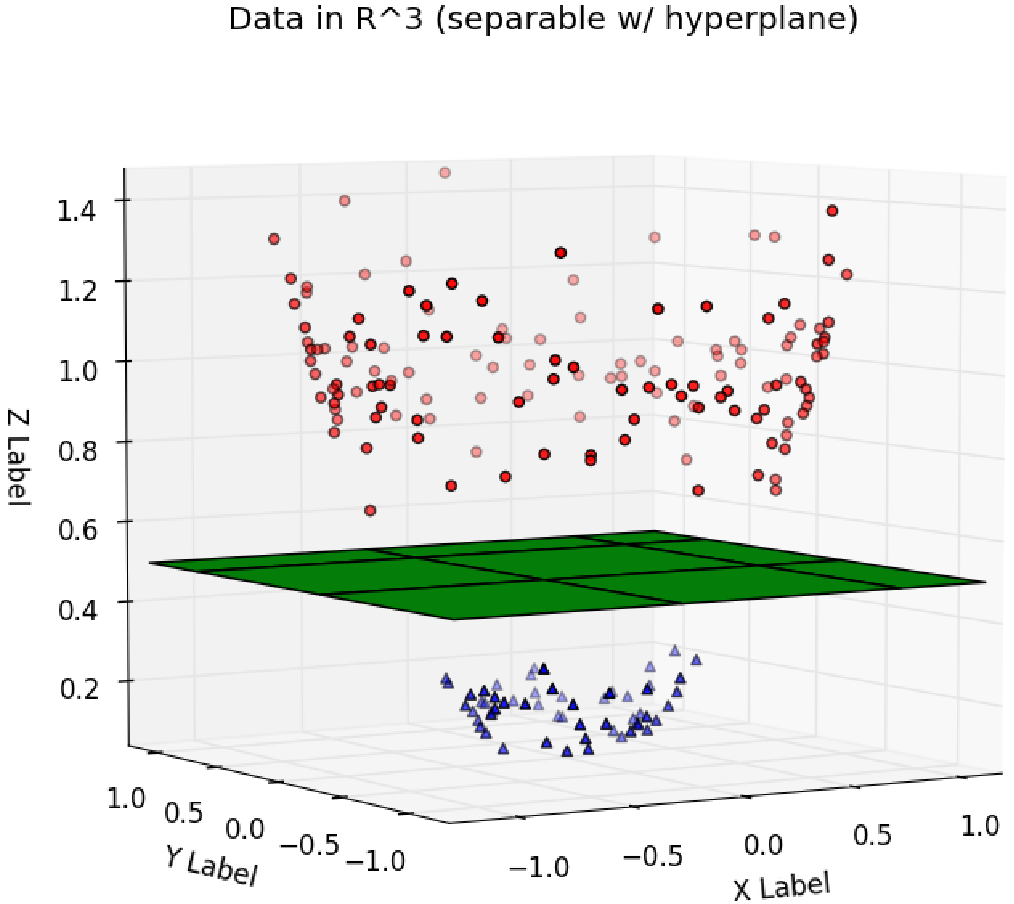
radial-basis functions:

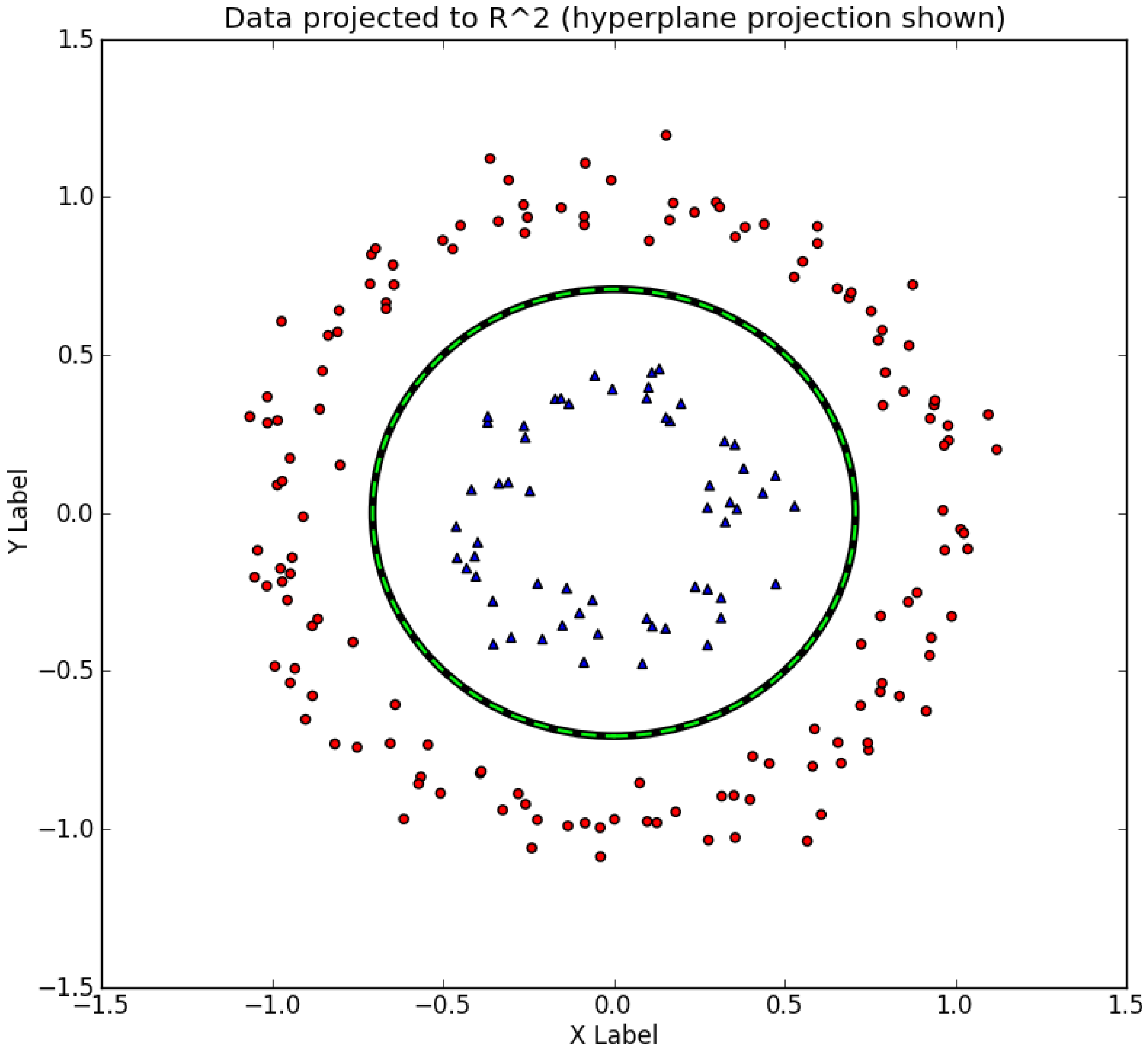
$$K(\boldsymbol{x}^t, \boldsymbol{x}) = \exp\left[-\frac{\|\boldsymbol{x}^t - \boldsymbol{x}\|^2}{2s^2}\right]$$

- Centro = x^t
- Radio = s

Gaussian Kernel for n-dimentions

$$G_{\mathrm{ND}}(\vec{x};\sigma) = \frac{1}{\left(\sqrt{2\pi}\,\sigma\right)^{N}}\, e^{-\frac{|\vec{x}|^2}{2\sigma^2}}$$


Data in R^3 (separable w/ hyperplane)
Z Label
Y Label
X Label


Data projected to R^2 (hyperplane projection shown)
Y Label
X Label

Implementación

- Scikitlearn
  - http://scikit-learn.org/stable/modules/svm.html
- Demo: https://github.com/Yagwar/TAMD/blob/master/Iris%20SVM%20clasifier.ipynb

Referencias

- Alpaydin (2010) Introduction to machine learning
- Manning, Raghavan & Schütze (2009) An Introduction to Information Retrieval
- Andrew Ng. Machine Learning course (MOOC) https://www.coursera.org/learn/machine-learning/home/welcome
- How SVM (Support Vector Machine) algorithm works https://www.youtube.com/watch?v=1NxnPkZM9bc
- Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011. http://scikit-learn.org/stable/