Aprenentatge Automàtic 2

GCED

Lluís A. Belanche

belanche@cs.upc.edu

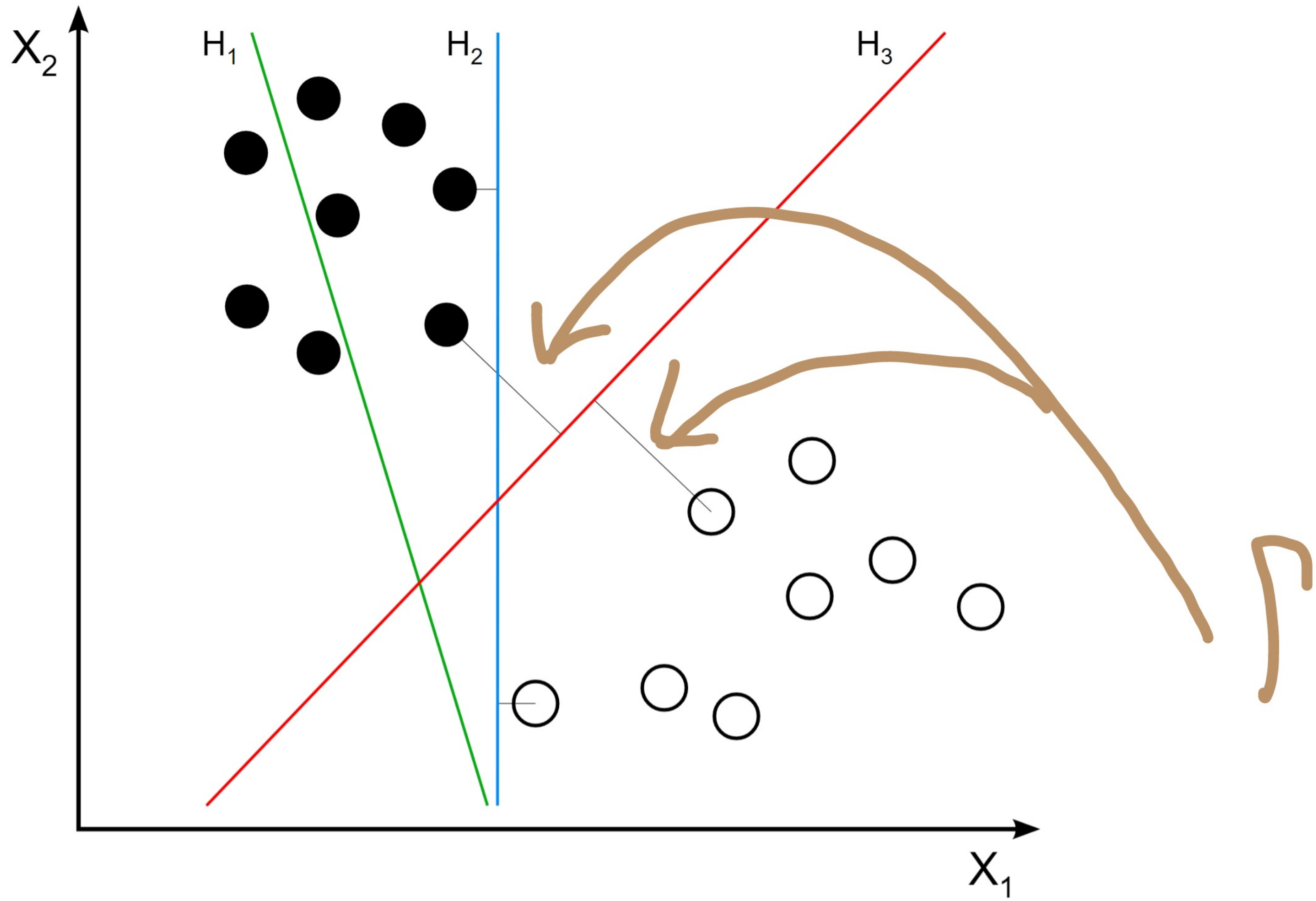
Soft Computing Research Group
Dept. de Ciències de la Computació (Computer Science)

Universitat Politècnica de Catalunya

2021-2022

LECTURE 3: From the Perceptron to the Suport vector machine (SVM). The SVM for classification. VC-dimension for the SVMC

Support Vector Machines



Motivation: hyperplanes with a larger (smaller) **margin** have reduced (increased) chances to separate the data ("complexity" is smaller (larger)).

Support Vector Machines

Formalisation reminder

Definition 1 *The functional margin* γ_i^F *of an example* $(\boldsymbol{x}_i, y_i)$ *wrt an hyperplane* $\boldsymbol{\omega}, b$ *is*

$$\gamma_i^F := y_i g(\boldsymbol{x}_i) = y_i(\boldsymbol{\omega}^\top \boldsymbol{x}_i + b)$$

We note that $\gamma_i^F > 0$ iff $(\boldsymbol{x}_i, y_i)$ is correctly classified by the hyperplane.

Definition 2 *The geometric margin* γ_i^G *of an example* $(\boldsymbol{x}_i, y_i)$ *is the functional margin wrt the hyperplane* $\frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|}, \frac{b}{\|\boldsymbol{\omega}\|}$.

We note that $\gamma_i^G = d(\boldsymbol{x}_i, \pi)$*, where* π *stands for the hyperplane* $\pi : \boldsymbol{\omega}^\top \boldsymbol{x} + b = 0$. ¶

Support Vector Machines

Formalisation reminder

Definition 3 *The margin Γ of a dataset D is the maximum geometric margin over all possible hyperplanes:*

$$\Gamma(D) := \sup_{(\boldsymbol{\omega},b)\in\mathbb{R}^m\times\mathbb{R}} \min_{i=1,\ldots,n} \gamma_i^G(\boldsymbol{\omega}, b)$$

Any hyperplane such that:

1. *it realises Γ on D*
2. *all its functional margins $\gamma_i^F > 0$*

is known as a **maximum margin hyperplane** *or MMH.*

Support Vector Machines

Formalisation

- We first set for the goal of learning a maximum margin hyperplane.

- It turns out that, if it exists, such a solution hyperplane is not unique (there is again an infinite number!)

- Rescaling $\boldsymbol{\omega}, b$ such that $|\boldsymbol{\omega}^\top \boldsymbol{x}_i + b| = 1$ for the data points closest to the hyperplane, we obtain $|\boldsymbol{\omega}^\top \boldsymbol{x}_i + b| \geq 1$ for all points

- The **support vectors** (SVs) are those data points $\{\boldsymbol{x}_i$ for which $|\boldsymbol{\omega}^\top \boldsymbol{x}_i + b| = 1\}$

- We introduce the loss function $L(y, \boldsymbol{\omega}^\top \boldsymbol{x}) := \text{máx}(1 - \gamma_i^F, 0)$ (called the **hinge loss**)

Support Vector Machines

Formalisation

- The **margin** becomes now twice the distance of any SV to the plane π:

$$2\,d(\boldsymbol{x}_{\mathsf{SV}}, \pi) = \frac{2}{\|\boldsymbol{\omega}\|}, \qquad \text{¶}$$

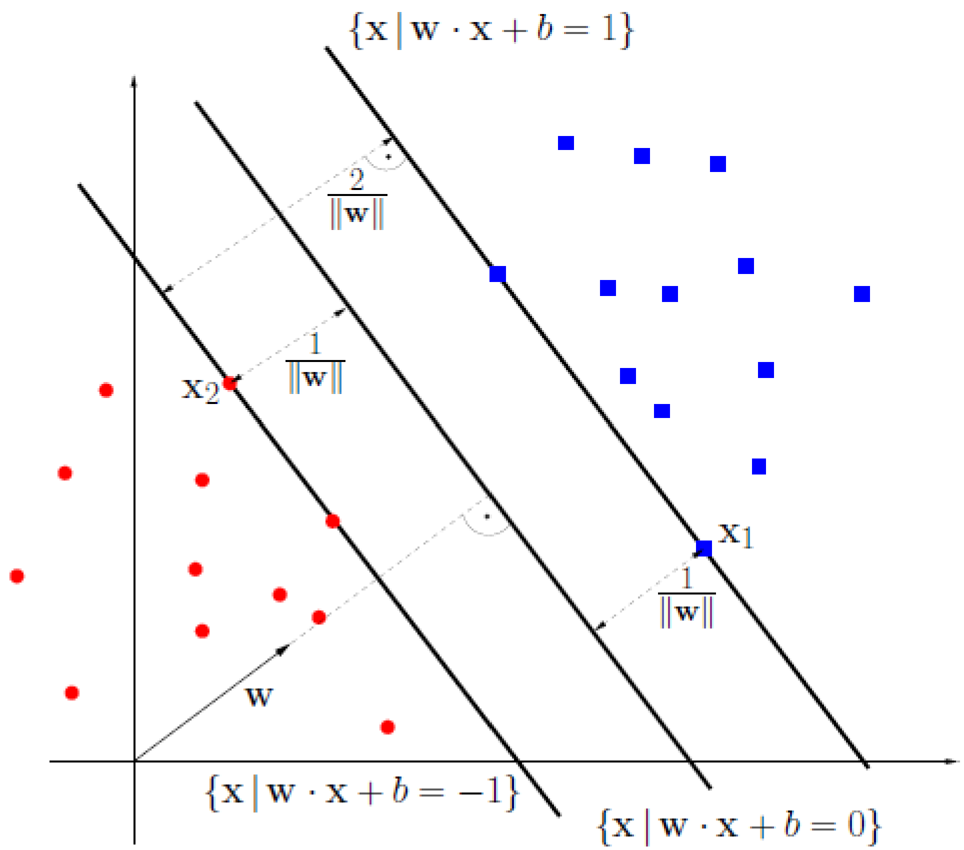
  since $|g(\boldsymbol{x}_{\mathsf{SV}})| = 1$.

- Therefore we can find the **canonical** MMH by solving

$$\max_{\boldsymbol{\omega}, b} \left\{ \frac{2}{\|\boldsymbol{\omega}\|} \;\;/\;\; y_i\,(\boldsymbol{\omega}^\top \boldsymbol{x}_i + b) \geq 1, \qquad 1 \leq i \leq n \right\}$$

Support Vector Machines

Geometrical view of the canonical MMH



Support Vector Machines

A look on what's to come

1. The solution for $\boldsymbol{\omega}$ can be expressed as $\boldsymbol{\omega} = \sum_{i=1}^{n} y_i \alpha_i \boldsymbol{x}_i$, $\alpha_i \geq 0$.

   (as a consequence of the versatile **Representer theorem**)

2. A fraction of the training data vectors will have $\alpha_i = 0$ (**sparsity**, as a consequence of the chosen **hinge loss** function)

3. The $\boldsymbol{x}_i$ for which $\alpha_i > 0$ will coincide with the **support vectors**

4. The SVM classifier is written

$$f_{\text{SVM}}(\boldsymbol{x}) = \text{sgn}(\boldsymbol{\omega}^\top \boldsymbol{x} + b) = \text{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i \boldsymbol{x}^\top \boldsymbol{x}_i + b\right)$$

Support Vector Machines

Formulation

$$\underset{\boldsymbol{\omega}, b}{\textbf{minimize}} \qquad \frac{1}{2}\|\boldsymbol{\omega}\|^2$$

$$\textbf{subject to } y_i\left(\boldsymbol{\omega}^\top \boldsymbol{x}_i + b\right) \geq 1, \qquad 1 \leq i \leq n$$

This is solved (numerically) by QP techniques:

- Quadratic (therefore convex) function subject to linear constraints
- Unique solution (or set of equivalent ones)
- Therefore, no local minima

Support Vector Machines

Formulation

For the set of constraints to be satisfied, the data set must be linsep; this is a very unrealistic requirement in practice ...

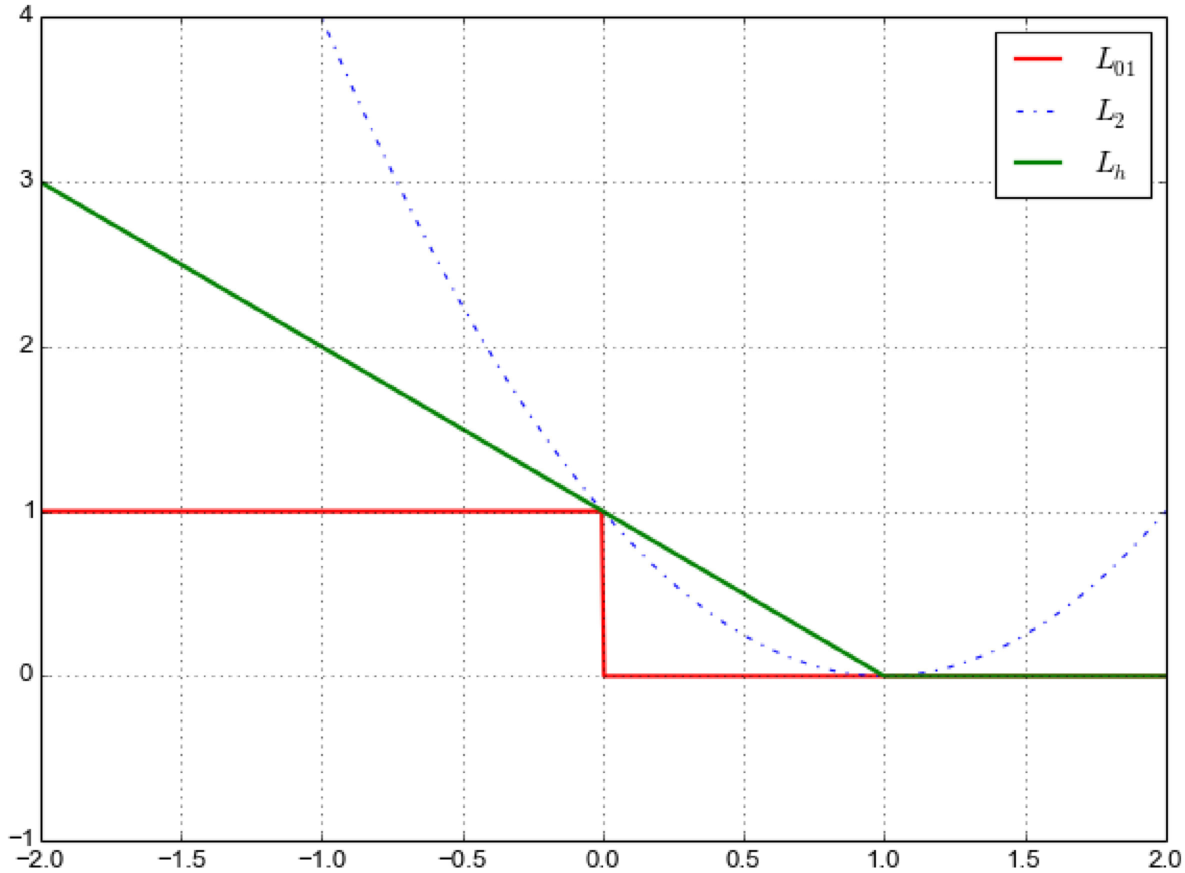
- We could aim at minimizing the **number** of violated constraints $|\{n \;\;/\;\; y_i\,(\boldsymbol{\omega}^\top \boldsymbol{x}_i + b) < 1\}|$, but this turns out to be NP-hard ...

- Instead, we can minimize the total **hinge loss**, a convex function of $\boldsymbol{\omega}$:

$$\underset{\boldsymbol{\omega},b}{\textbf{minimize}} \qquad \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\sum_{i=1}^{n} \text{máx}(1-\gamma_i^F, 0), \qquad C > 0$$

Support Vector Machines

Formulation



L_{01} is the 0/1 loss; L_2 is the square loss; L_h is the hinge loss

Support Vector Machines

Margin violations

- This problem may be rewritten as another QP, by introducing a set of margin violations ε_i –called **slack** variables in optimization–, for each $\boldsymbol{x}_i$:

$$\underset{\boldsymbol{\omega}, b, \{\varepsilon_i\}}{\textbf{minimize}} \qquad \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\sum_{i=1}^{n} \varepsilon_i$$

$$\textbf{subject to } y_i\,(\boldsymbol{\omega}^\top \boldsymbol{x}_i + b) \geq 1 - \varepsilon_i \textbf{ and } \varepsilon_i \geq 0 \ (1 \leq i \leq n)$$

- This is a **soft** margin ($\varepsilon_i > 0$ implying $\boldsymbol{x}_i$ would violate the original constraint)
- For a training error to occur, $\varepsilon_i > 1$ and so $\sum_{i=1}^{n} \varepsilon_i$ is an upper bound on the number of training errors
- The optimal slacks satisfy $\varepsilon_i = \text{máx}(1 - \gamma_i^F, 0)$

Support Vector Machines

SVM Lagrangian (primal)

We now construct the **Lagrangian**:

$$\mathcal{L} = \frac{1}{2}\|\boldsymbol{\omega}\|^2 - \sum_{i=1}^{n} \alpha_i \Big\{ y_i \left(\boldsymbol{\omega}^\top \boldsymbol{x}_i + b\right) - 1 + \varepsilon_i \Big\} + C \sum_{i=1}^{n} \varepsilon_i - \sum_{i=1}^{n} \mu_i \varepsilon_i$$

- The $\alpha_i, \mu_i \geq 0$ are the **Lagrange multipliers**; the μ_i ensure that $\varepsilon_i \geq 0$
- The solution is a **saddle point** of $\mathcal{L}$: minimum w.r.t. $\boldsymbol{\omega}, b$ and the ε_i and maximum w.r.t. the α_i and μ_i

Support Vector Machines

Lagrangian form

The gradient of $\mathcal{L}$ with respect to $\boldsymbol{\omega}, b$ and ε_i must vanish:

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{i=1}^{n} \alpha_i y_i = 0, \qquad \frac{\partial \mathcal{L}}{\partial \boldsymbol{\omega}} = \boldsymbol{\omega} - \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i = 0, \qquad \frac{\partial \mathcal{L}}{\partial \varepsilon_i} = C - \alpha_i - \mu_i = 0$$

In addition, the KKT complementarity conditions must hold:

$$\alpha_i \Big(y_i \, (\boldsymbol{\omega}^\top \boldsymbol{x}_i + b) - 1 + \varepsilon_i \Big) = 0 \qquad (1 \leq i \leq n)$$

Support Vector Machines

Dual formulation

The Lagrangian $\mathcal{L}$ is convex; its optimization is equivalent to the maximization of its concave **dual problem** $\mathcal{L}_D$:

$$\underset{\boldsymbol{\omega}, b, \{\alpha_i\}}{\textbf{minimize}} \qquad \mathcal{L}_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \, \boldsymbol{x}_i{}^{\top} \boldsymbol{x}_j$$

$$\textbf{subject to } 0 \leq \alpha_i \leq C \ (1 \leq i \leq n), \qquad \textbf{and} \ \sum_{i=1}^{n} \alpha_i y_i = 0$$

- Neither $\mu_i, \varepsilon_i, \boldsymbol{\omega}, b$ appear in the dual form; maximization is only w.r.t. the α_i
- This optimization problem is expressed *only* in terms of inner products of the data points: the dual lends itself to kernelisation
- How many free parameters? n (independent of data dimension)

Support Vector Machines

Dual formulation

A closer look at the KKT complementarity conditions:

- $\alpha_i = 0$ implies $y_i\, g(\boldsymbol{x}_i) > 1$ and $\varepsilon_i = 0$ ($\boldsymbol{x}_i$ is **not a SV**)

- $\alpha_i \in (0, C)$ implies $y_i\, g(\boldsymbol{x}_i) = 1$ and $\varepsilon_i = 0$ ($\boldsymbol{x}_i$ is a **non-bound SV**)

- $\alpha_i = C$ implies $y_i\; g(\boldsymbol{x}_i) < 1$ and $\varepsilon_i > 0$ ($\boldsymbol{x}_i$ is a **bound SV**)

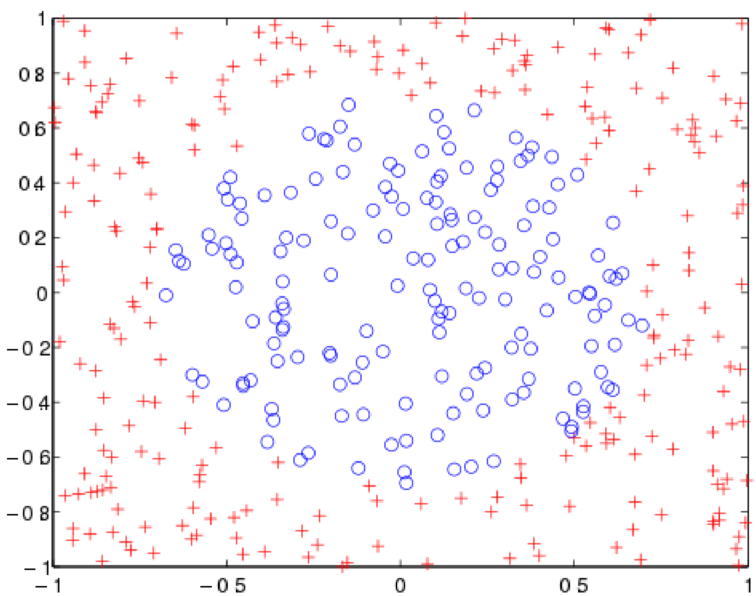
  (in particular, $\varepsilon_i > 1$ implies $\boldsymbol{x}_i$ is a **training error**)

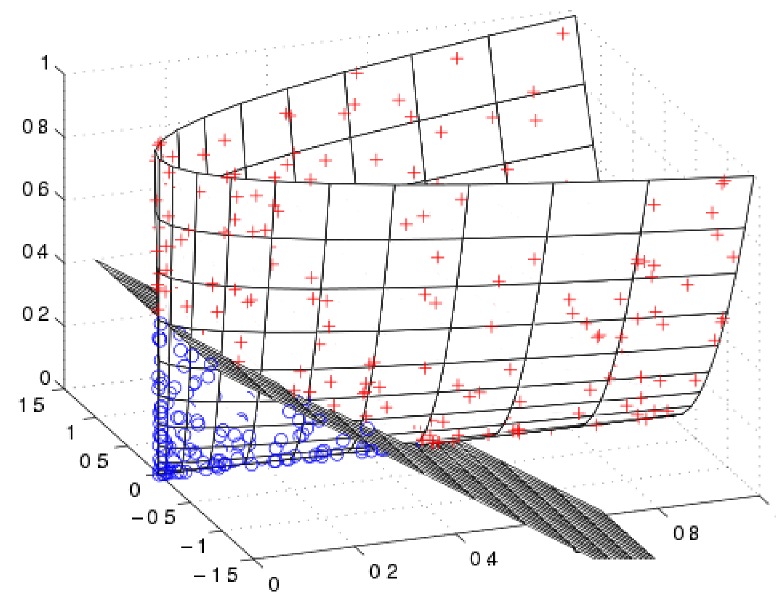
Support Vector Machines

The SVM goes non-linear

Recall the idea of mapping input data into some Hilbert space (called the **feature space**) via a non-linear mapping $\phi : \mathcal{X} \to \mathcal{H}$

The associated kernel function is $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle_{\mathcal{H}}\,,\ \ \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$





Support Vector Machines

SVM kernelization

- We now substitute $\boldsymbol{x}_i$ by $\phi(\boldsymbol{x}_i)$, then build the MMH in $\mathcal{H}$
- The dual of the new QP problem is formulated exactly as before, replacing ${\boldsymbol{x}_i}^\top \boldsymbol{x}_j$ with $\phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j)_{\mathcal{H}} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$
- The final SVM classifier becomes:

$$f_{\mathsf{SVM}}(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i k(\boldsymbol{x}, \boldsymbol{x}_i) + b\right)$$

Support Vector Machines

LOO bounds (II)

Theorem 1 *The LOOCV error of a stable SVM*$^{(*)}$ *on a set of training patterns* $\boldsymbol{x}_i$ *is bounded by* $|\{i\,/\,(2\alpha_i R^2 + \varepsilon_i) \geq 1\}|/n$, *where* R^2 *is an upper bound on* $k(\boldsymbol{x}, \boldsymbol{x})$ *and* $k(\boldsymbol{x}, \boldsymbol{x}') \geq 0$.

- This quantity can be extracted easily from the solution
- This LOOCV error is an unbiased estimate of true error

$^{(*)}$ A SVM is stable if there is at least one non-bound SV (see *Estimating the Generalization Performance of a SVM Efficiently*. T. Joachims; In ICML, 2000)

Support Vector Machines

Final remarks (I)

- The fact that the **MMH** is determined only by the support vectors is most remarkable, since usually this number will be usually small

- The **support vectors** (SVs) are:

  1. the only training examples that define the solution

  2. the most difficult examples to classify

- This means all the **relevant information** in the data set is summarized by the SVs: we would have obtained the same result by using *only* the SVs from the outset

Support Vector Machines

Final remarks (II)

- The SVM is specially well suited for "large m, low n" problems, because:

  1. complexity grows with n (non-parametric model)

  2. space requirements (the kernel matrix) also grows with n

  3. generalization error does not depend on m (theoretically)

- The "architecture" is determined automatically by the method (not by experimentation, as in neural networks)

Support Vector Machines

VC dimension

For a two-class classifier, the **VC dimension** ϑ is the maximum number of points that can be separated in all possible 2^ϑ ways (**shattered**) by using functions representable by the classifier.

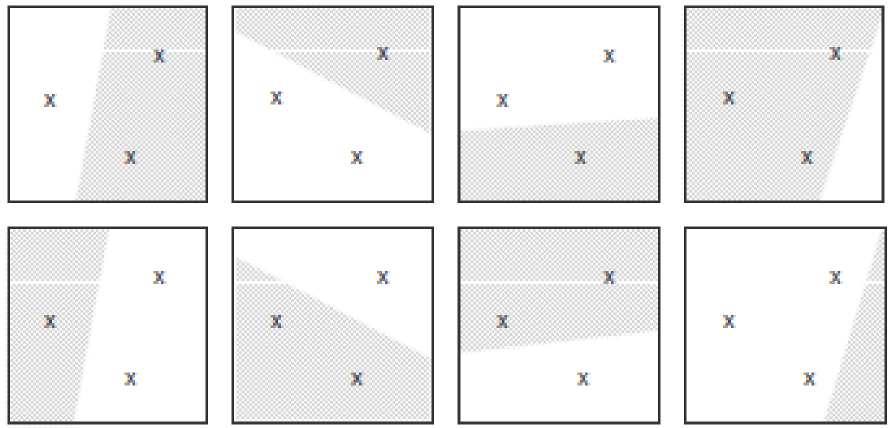
- Note it is *sufficient* that one set of ϑ points exists that can be shattered for the VC dimension to be at least ϑ

- If the VC dimension of a class is ϑ, this means there is at least one set of ϑ points that can be shattered by members of the class. It does not mean that every set of ϑ points can be shattered

- If no set of $\vartheta + 1$ points can be shattered by members of the class, then the VC dimension of the class is at most ϑ

Support Vector Machines

A basic example



- In $\mathbb{R}^2$ we can shatter these three points (VC dim is ≥ 3)

- No set of four or more points can be shattered (VC dim is < 4)

Support Vector Machines

Why is the VC dimension relevant?

Theorem 2 *(Vapnik and Chervonenkis, 1974). Let D be an i.i.d data sample of size n and $\mathcal{Y}$ a class of parametric binary classifiers. Let ϑ denote the VC dimension of $\mathcal{Y}$. Take $y \in \mathcal{Y}$ with empirical error $R_n(y)$ on D. For all $\eta > 0$ it holds true that, with probability at least $1 - \eta$, the true error of y is bounded by:*

$$R(y) \leq R_n(y) + H(n, \vartheta, \eta)$$

where

$$H(n, \vartheta, \eta) := \sqrt{\frac{\vartheta(\ln(2n/\vartheta) + 1) - \ln(\eta/4)}{n}}$$

Support Vector Machines

More than an intuition

- Separating hyperplanes in $\mathbb{R}^d$ have VC dimension $d + 1$
- When we use a feature map into a very high dimension $D \in (\mathbb{N} \cup \{\infty\})$, VC dimension will grow accordingly
- If we bound the margin of the hyperplanes, we limit VC dimension (therefore, we have an explicit control on complexity)

Support Vector Machines

More than an intuition

Theorem 3 *Consider canonical hyperplanes* $f(\boldsymbol{x}) = \mathrm{sgn}(\boldsymbol{\omega}^\top \boldsymbol{x} + b)$ *and a data set* $D = \{(\boldsymbol{x}_1, t_1), \ldots, (\boldsymbol{x}_n, t_n)\}$, *with* $\boldsymbol{x}_i \in \mathbb{R}^m$ *and* $y_i \in \{-1, +1\}$. *The* **subclass** *of linear classifiers with margin* μ *has VC dimension* ϑ *bounded by*

$$\vartheta \leq \text{mín}\left(\left\lceil \frac{R^2}{\mu^2} \right\rceil, m\right) + 1$$

where R *is the radius of the smallest sphere centered at the origin containing the* $\boldsymbol{x}_i$.