Machine Learning Fundamentals - LS2026
Support Vector Machines

Czech Technical University in Prague
V. Franc

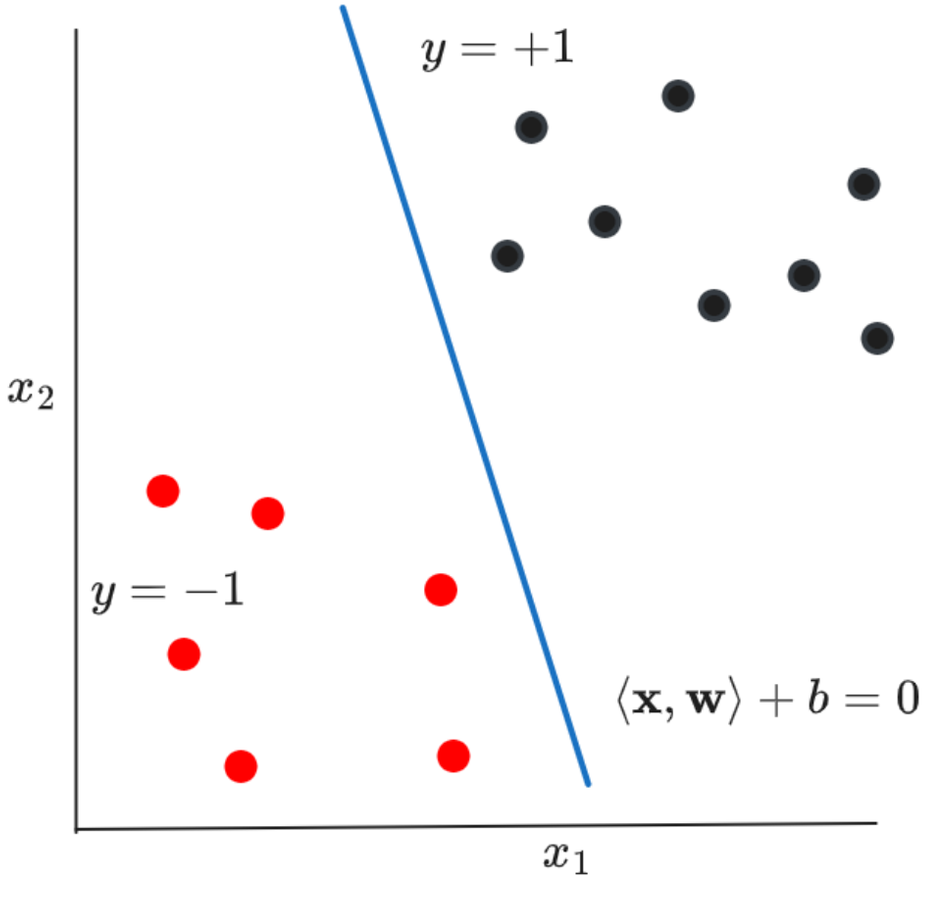
Linear classifier

- Assume linearly separable training data:

$$T_m = \left((\mathbf{x}, y) \in \mathbb{R}^d \times \{-1, +1\} \mid i = 1, \ldots, m\right)$$

- **Linear classifier:** $h\colon \mathbb{R}^d \to \{-1, +1\}$:

$$h(\mathbf{x}; \mathbf{w}, b) = \begin{cases} +1 & \text{if} \quad \langle \mathbf{x}, \mathbf{w} \rangle + b \geq 0 \\ -1 & \text{if} \quad \langle \mathbf{x}, \mathbf{w} \rangle + b < 0 \end{cases}$$



- **Empirical Risk Minimization**: find a predictor with the minimal error on training examples.
- In the linearly separable case, ERM amounts to finding a linear classifier $(\mathbf{w}, b)$ with zero training error:

$$\begin{aligned} \langle \mathbf{x}_i, \mathbf{w} \rangle + b &> 0\,, \quad i \in \mathcal{I}_+ \\ \langle \mathbf{x}_i, \mathbf{w} \rangle + b &< 0\,, \quad i \in \mathcal{I}_- \end{aligned}$$

where $\mathcal{I}_+ = \{i \in \{1, \ldots, m\} \mid y_i = +1\}$, and $\mathcal{I}_- = \{i \in \{1, \ldots, m\} \mid y_i = -1\}$ denote the indices of positive and negative examples, respectively.

- **Remark:** For linearly separable data, there are infinitely many linear classifiers with zero training error. From the ERM point of view, they are equivalent on the training set. Will they all generalize equally well to unseen data?

Maximum Margin Classifier

- Assume linearly separable training data:

$$T_m = \left((\mathbf{x}, y) \in \mathbb{R}^d \times \{-1, +1\} \mid i = 1, \ldots, m\right)$$

- **Linear classifier:** $h\colon \mathbb{R}^d \to \{-1, +1\}$:

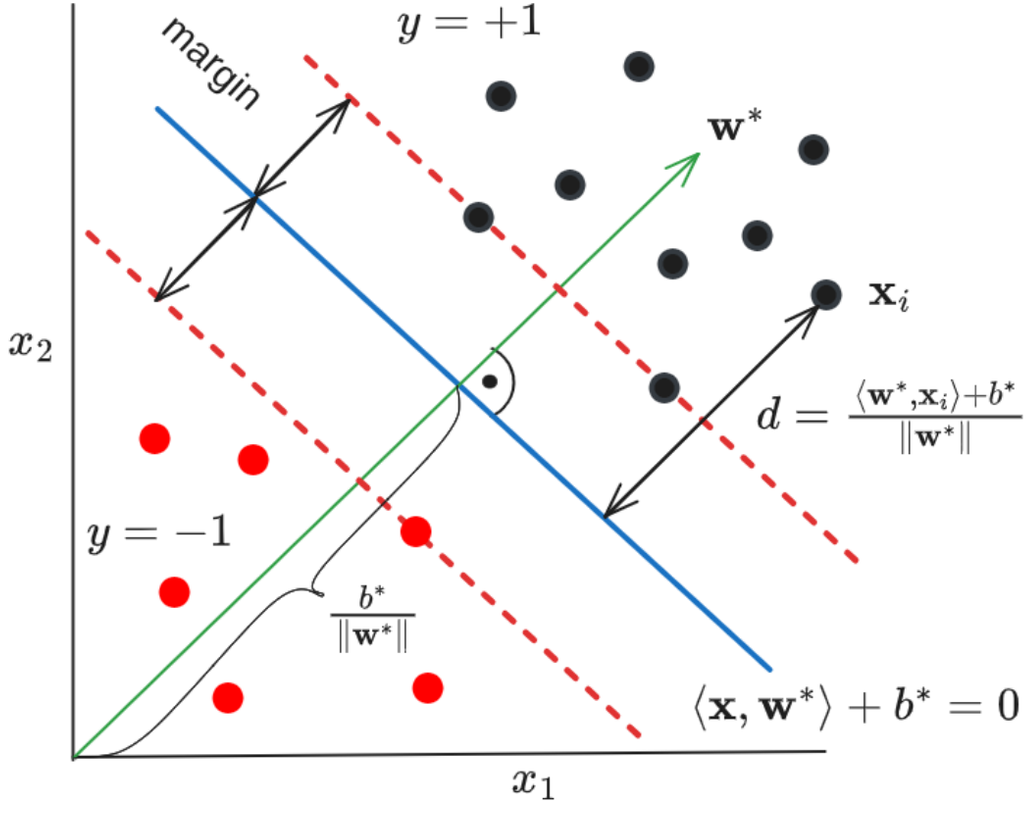
$$h(\mathbf{x}; \mathbf{w}, b) = \begin{cases} +1 & \text{if} \quad \langle \mathbf{x}, \mathbf{w} \rangle + b \geq 0 \\ -1 & \text{if} \quad \langle \mathbf{x}, \mathbf{w} \rangle + b < 0 \end{cases}$$

- Signed distance between $\mathbf{x}$ and $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$:

$$d = \frac{\langle \mathbf{w}, \mathbf{x} \rangle + b}{\|\mathbf{w}\|}$$

- **Margin:** minimal distance between the hyperplane $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$ and data T_m:

$$\gamma(\mathbf{w}, b) = \min_{i=1,\ldots,m} y_i \frac{\langle \mathbf{w}, \mathbf{x}_i \rangle + b}{\|\mathbf{w}\|}$$



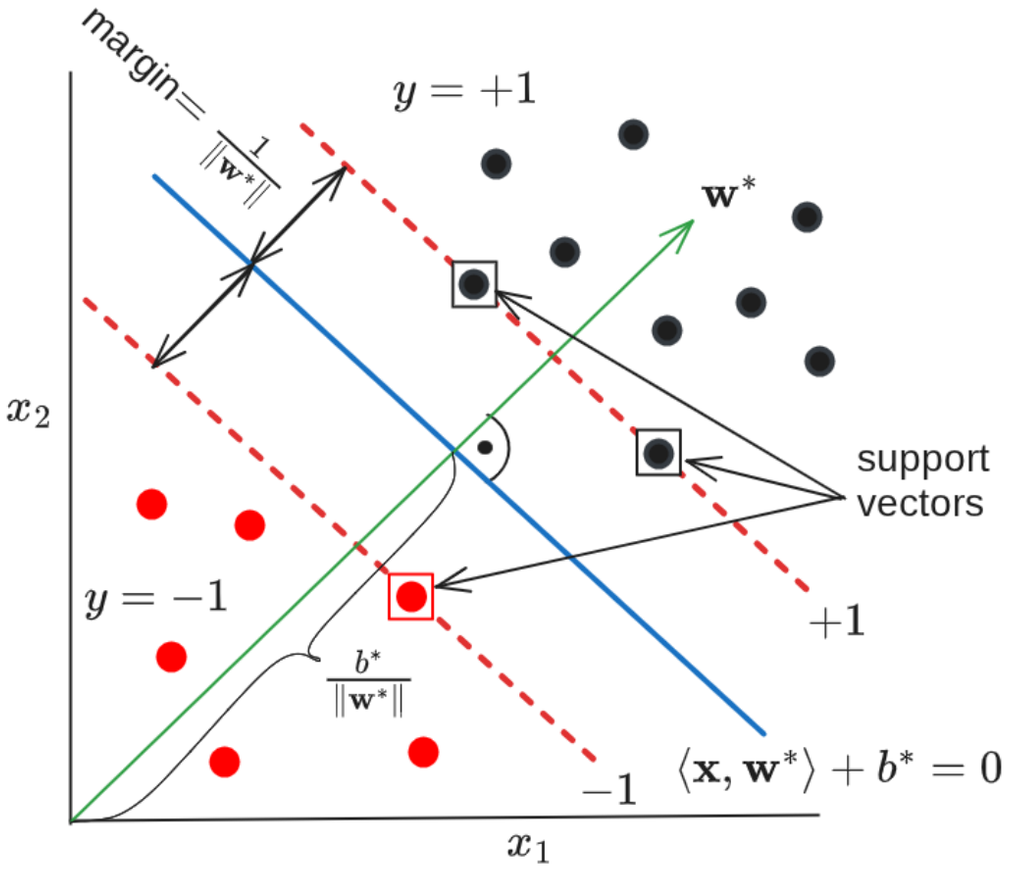
- **Maximum margin classifier:** linear classifier with the maximum margin:

$$(\mathbf{w}^*, b^*) = \operatorname*{arg\,max}_{\mathbf{w} \in \mathbb{R}^d \setminus \{\mathbf{0}\}, b \in \mathbb{R}} \left[\min_{i=1,\ldots,m} y_i \frac{\langle \mathbf{w}, \mathbf{x}_i \rangle + b}{\|\mathbf{w}\|} \right]$$

Hard-margin Support Vector Machines

- **Maximum margin classifier:** linear classifier with the maximum margin:

$$(\mathbf{w}^*, b^*) = \underset{\mathbf{w}\in\mathbb{R}^d\setminus\{\mathbf{0}\}, b\in\mathbb{R}}{\arg\max} \left[\min_{i=1,\dots,m} y_i \frac{\langle \mathbf{w}, \mathbf{x}_i\rangle + b}{\|\mathbf{w}\|} \right]$$



- It can be expressed as **Quadratic Program**:

$$(\mathbf{w}^*, b^*) = \underset{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}}{\arg\min} \frac{1}{2}\|\mathbf{w}\|^2$$

subject to

$$\begin{aligned} \langle \mathbf{w}, \mathbf{x}_i\rangle + b &\geq +1\,, \quad i \in \mathcal{I}_+ \\ \langle \mathbf{w}, \mathbf{x}_i\rangle + b &\leq -1\,, \quad i \in \mathcal{I}_- \end{aligned}$$

- **Support Vectors:** training examples that are nearest to the decision hyperplane

$$\mathcal{I}_{\mathrm{SV}} = \{i \in \{1,\dots,m\} \mid y_i \langle \mathbf{w}^*, \mathbf{x}_i\rangle + b^* = 1\}$$

Remark: Removing the support vectors $\mathcal{I}_{\mathrm{SV}}$ from the training data T_m does not change the solution.

Soft-margin Support Vector Machines

- Assume possibly **non-separable training data**:

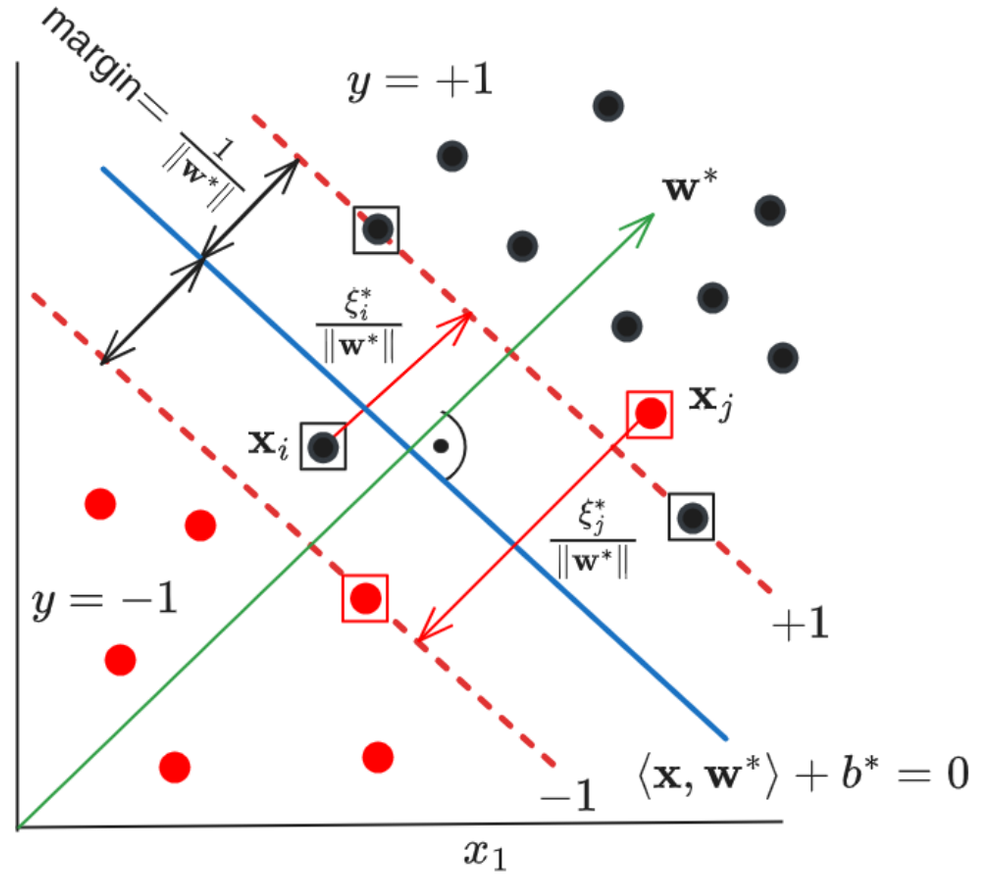
$$T_m = \left((\mathbf{x}, y) \in \mathbb{R}^d \times \{-1, +1\} \mid i = 1, \ldots, m\right)$$

- The Soft-Margin SVM classifier is obtained as the solution to the following **Quadratic Program**:

$$(\mathbf{w}^*, b^*, \xi^*) = \underset{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}, \xi\in\mathbb{R}^m}{\arg\min} \left[\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i\right]$$

subject to

$$\begin{aligned} \langle \mathbf{w}, \mathbf{x}_i\rangle + b &\geq +1 - \xi_i\,, & i \in \mathcal{I}_+ \\ \langle \mathbf{w}, \mathbf{x}_i\rangle + b &\leq -1 + \xi_i\,, & i \in \mathcal{I}_- \\ \xi_i &\geq 0\,, & i \in \mathcal{I}_+ \cup \mathcal{I}_- \end{aligned}$$



- **Slack variables** $\xi = (\xi_1, \ldots, \xi_m)$**:** ensure feasibility of the constraints for non-separable data. Each ξ_i approximates the loss on the example $(\mathbf{x}_i, y_i)$ because $\xi_i \geq [\![h(\mathbf{x}_i; \mathbf{w}, b) \neq y_i]\!]$.
- **Regularization constant** $C > 0$**:** a hyperparameter (not learned from data) controlling the trade-off between minimizing the empirical error proxy $\sum_i \xi_i$ and maximizing the margin (minimizing $\frac{1}{2}\|\mathbf{w}\|^2$).
- **Support Vectors:** training examples on or inside the margin determining the decision hyperplane $\mathcal{I}_{\text{SV}} = \{i \in \{1, \ldots, m\} \mid y_i\langle \mathbf{w}^*, \mathbf{x}_i\rangle + b^* \leq 1\}$.

SVMs implement Regularized Risk Minimization principle

- Learning a soft-margin SVM classifier amounts to solving the following quadratic program:

$$(\mathbf{w}^*, b^*, \xi^*) = \underset{\mathbf{w}\in\mathbb{R}^d,\, b\in\mathbb{R},\, \xi\in\mathbb{R}^m}{\arg\min} \left[\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i\right]$$

$$\begin{array}{rrcll} & \langle \mathbf{w}, \mathbf{x}_i\rangle + b & \geq & 1-\xi_i\,, & i\in\mathcal{I}_+ \\ \text{s.t.} & \langle \mathbf{w}, \mathbf{x}_i\rangle + b & \leq & -1+\xi_i\,, & i\in\mathcal{I}_- \\ & \xi_i & \geq & 0\,, & i=1,\ldots,m \end{array}$$

- This quadratic program can be rewritten as a **Regularized Risk Minimization** problem:

$$\begin{aligned} (\mathbf{w}^*, b^*) &= \underset{\mathbf{w}\in\mathbb{R}^d,\, b\in\mathbb{R}}{\arg\min} \left[\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\max\{0,\ 1-y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b)\}\right] \\ &= \underset{\mathbf{w}\in\mathbb{R}^d,\, b\in\mathbb{R}}{\arg\min} \left[\lambda\,\Omega(\mathbf{w}, b) + \hat{R}_{\text{hinge}}(T_m, \mathbf{w}, b)\right] \end{aligned}$$

- where
  - $\Omega(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2$ is the quadratic **regularization term**,
  - $\hat{R}_{\text{hinge}}(T_m, \mathbf{w}, b) = \frac{1}{m}\sum_{i=1}^{m}\psi\big(y_i, f(\mathbf{x}_i, \mathbf{w}, b)\big)$ is the **empirical risk** under the hinge loss

$$\psi(y, t) = \max\{0,\ 1-y\,t\},$$

    where $f(\mathbf{x}, \mathbf{w}, b) = \langle \mathbf{w}, \mathbf{x}\rangle + b$ is the decision score of the linear predictor.
  - $\lambda = \frac{1}{2\,m\,C}$ is the **regularization parameter**.

Hinge loss

- The linear classifier:

$$h(\mathbf{x}; \mathbf{w}, b) = \operatorname{sign}\big(f(\mathbf{x}; \mathbf{w}, b)\big), \qquad f(\mathbf{x}; \mathbf{w}, b) = \langle \mathbf{w}, \mathbf{x} \rangle + b.$$

- The empirical risk under the hinge loss:

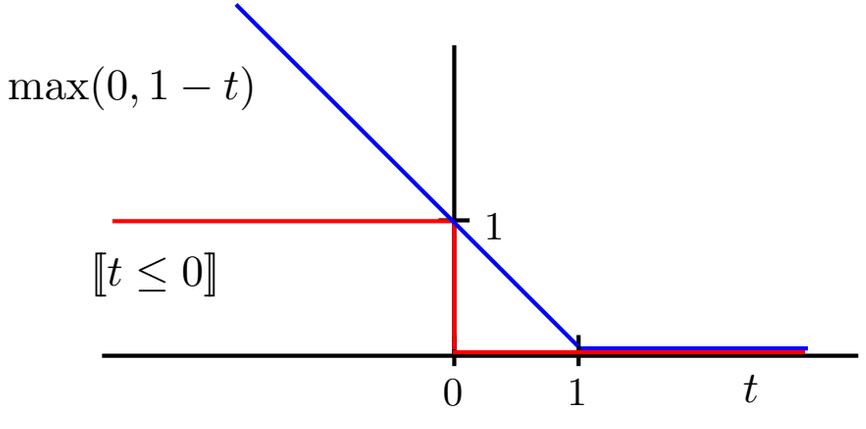
$$\hat{R}_{\text{hinge}}(T_m, \mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \psi\big(y_i, f(\mathbf{x}_i, \mathbf{w}, b)\big), \qquad \psi(y, t) = \max\{0,\, 1 - y\, t\},$$

  a convex upper bound on the empirical risk under the target $0/1$ loss:

$$\hat{R}_{0/1}(T_m, \mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^{m} [\![y_i \neq h(\mathbf{x}_i; \mathbf{w}, b)]\!]$$

- The hinge loss is a **convex upper bound** on the $0/1$ loss used to penalize predictions of a linear classifier:

$$\underbrace{[\![\operatorname{sign}(f(\mathbf{x})) \neq y]\!]}_{0/1 \text{ loss}} = [\![y\, f(\mathbf{x}) \leq 0]\!] \leq \underbrace{\max\{0,\, 1 - y\, f(\mathbf{x})\}}_{\psi(y, f(\mathbf{x}))}$$



SVM implement Structured Risk Minimization

Strutural Risk Minimization:

- Let $\mathcal{H} = \{-1, +1\}^{\mathcal{X}}$, then for all $h \in \mathcal{H}$ with probability $1 - \delta$ at least over i.i.d. generated T_m:

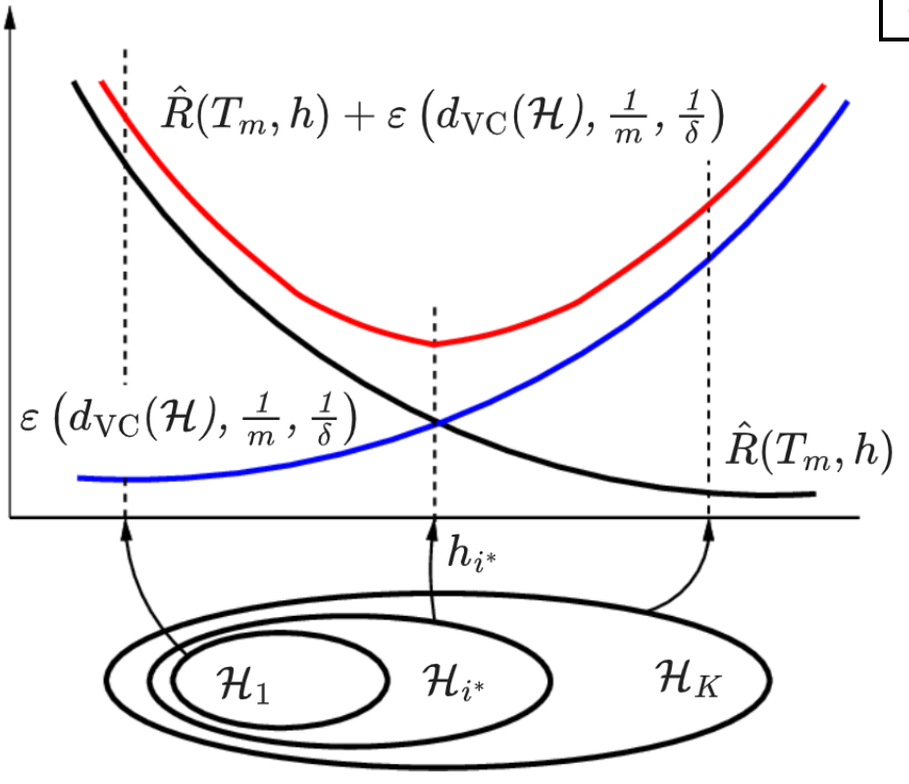
$$R(p, h) \leq \hat{R}(T_m, h) + \varepsilon\left(\mathrm{VCdim}(\mathcal{H}), \frac{1}{m}, \frac{1}{\delta}\right)$$

- Structure of nested sequence of hypothesis classes:

$$\mathcal{H}_1 \subset \mathcal{H}_2 \subset \cdots \subset \mathcal{H}_K$$

- For each $i \in \{1, \ldots, K\}$, apply ERM:

$$h_i = \arg\min_{h \in \mathcal{H}_i} \hat{R}(T_m, h)$$

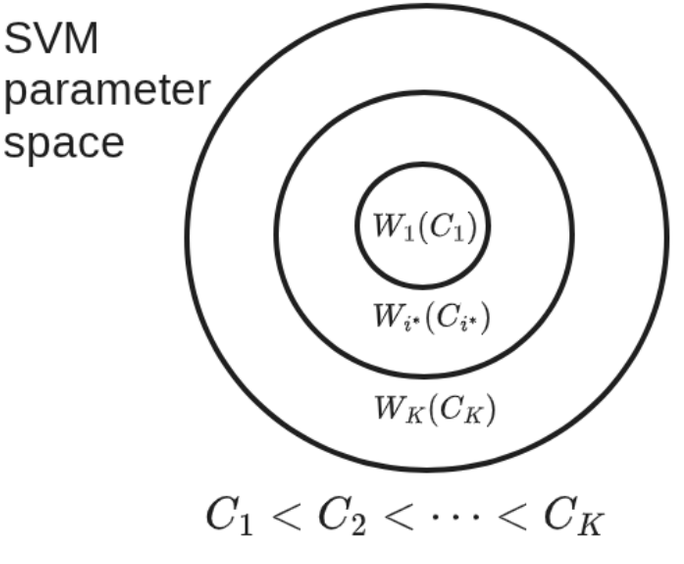


Support Vector Machines:

- Let $\mathcal{H} = \{h(\mathbf{x}) = \mathrm{sign}(\langle \mathbf{w}, \mathbf{x} \rangle) \mid \|\mathbf{w}\| \leq W\}$, then for all $h \in \mathcal{H}$ with probality $1 - \delta$ at least over i.i.d. generated T_m:

$$R(p, h) \leq \hat{R}_{\mathrm{hinge}}(T_m, h) + \varepsilon_{\mathrm{SVM}}\left(W, \frac{1}{m}, \frac{1}{\delta}\right)$$

where $\hat{R}_{\mathrm{hinge}}(T_m, h) = \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i \langle \mathbf{w}, \mathbf{x}_i \rangle\}$.



Non-linear feature mapping

- A linear classifier in a nonlinear feature space can represent nonlinear decision boundaries:

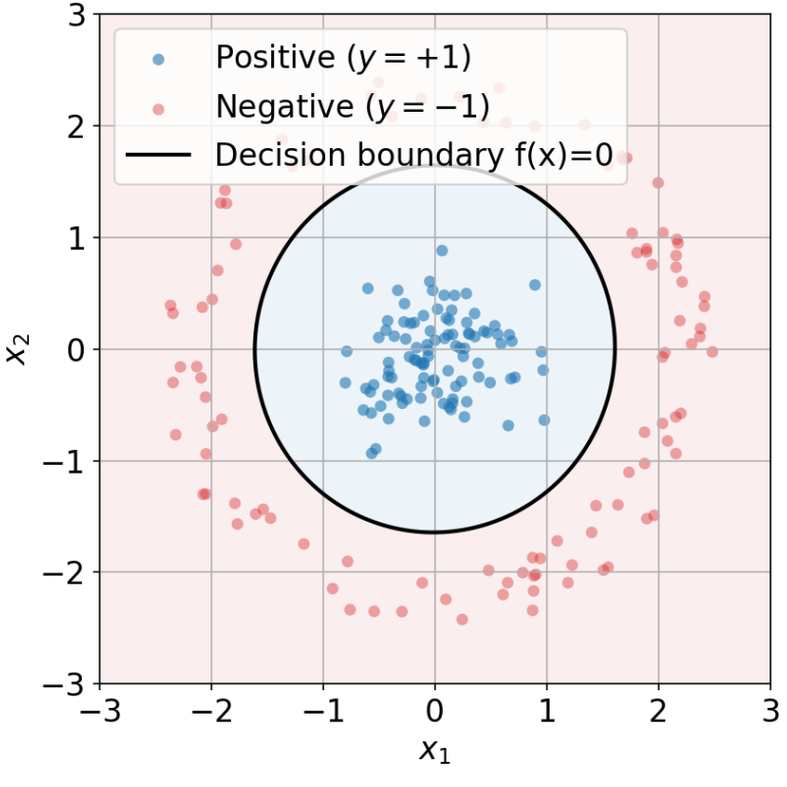
$$h(x; \mathbf{w}, b) = \text{sign}\left(w_1\phi_1(x) + \cdots + w_n\phi_n(x) + b\right) = \text{sign}\left(\langle \mathbf{w}, \phi(x)\rangle + b\right)$$

- Kernel functions allow us to compute inner products in the mapped feature space efficiently:

$$k(x, x') = \langle \phi(x), \phi(x')\rangle$$

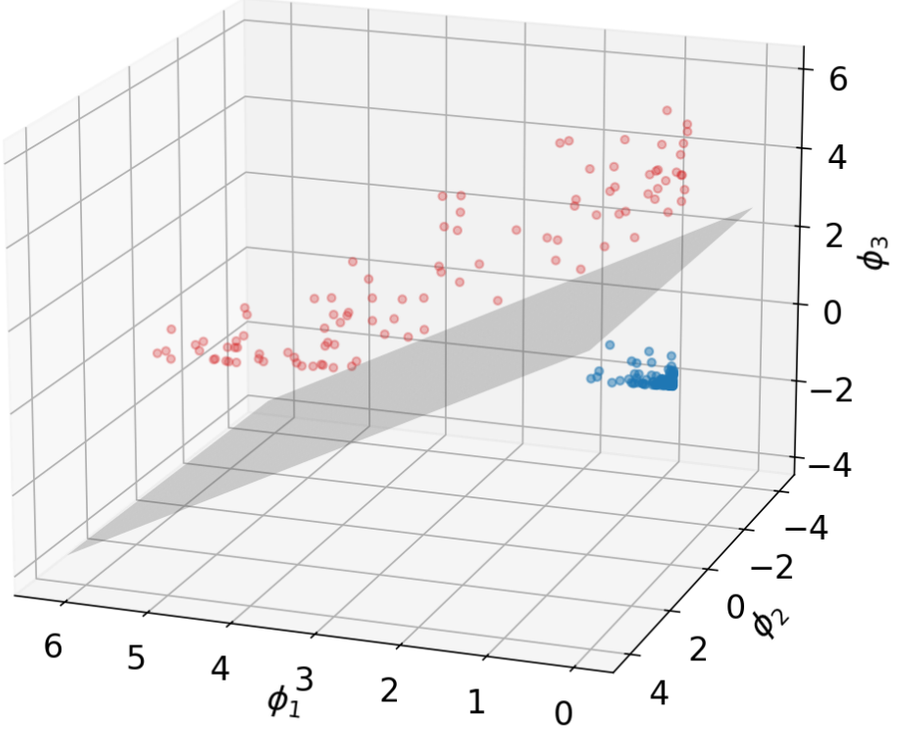
Original features

$\mathbf{x} = [x_1, x_2] \in \mathbb{R}^2$



New features

$\phi(\mathbf{x}) = [x_1^2, \sqrt{2}x_1x_2, x_2^2] \in \mathbb{R}^3$



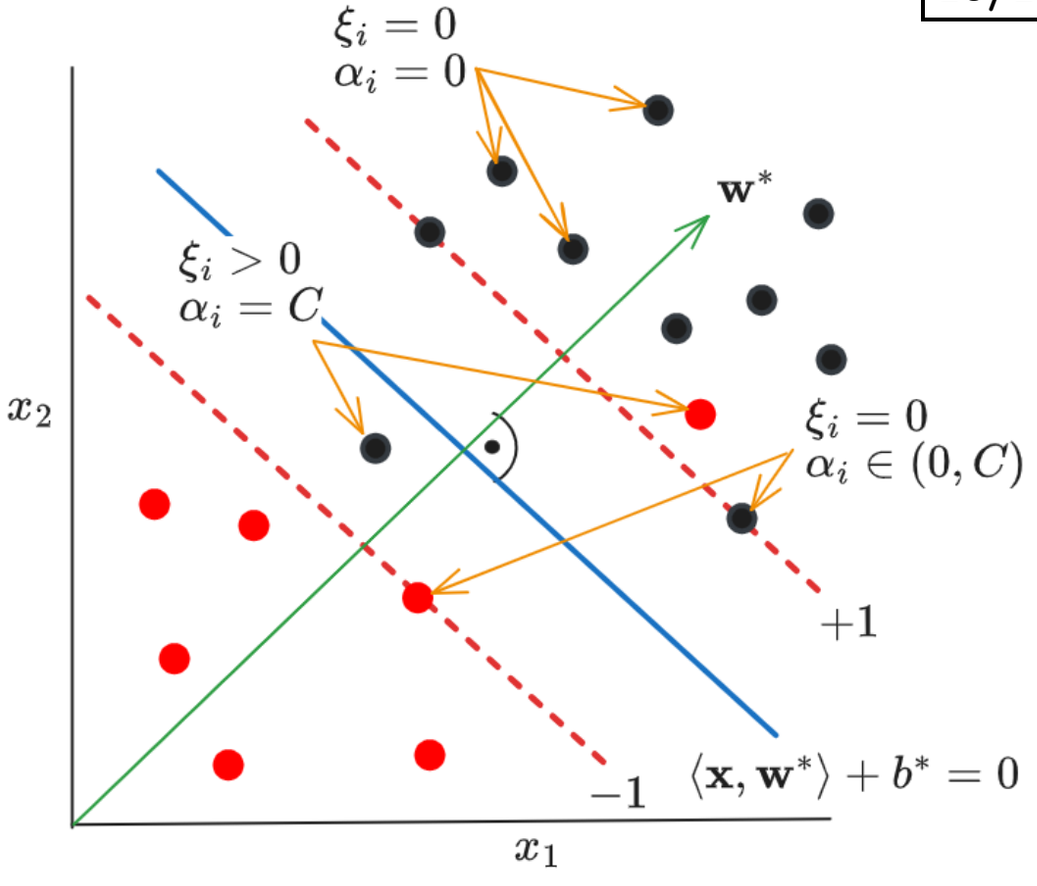
Primal and dual SVM Quadratic Program

- **Primal QP:** ($d+1+m$ vars; $2m$ constr)

$$(\mathbf{w}^*, b^*, \xi^*) = \underset{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}, \xi\in\mathbb{R}^m}{\arg\min} \left[\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i\right]$$

subject to

$$\begin{aligned} \langle \mathbf{w}, \mathbf{x}_i\rangle + b &\geq +1+\xi_i\,, & i\in\mathcal{I}_+ \\ \langle \mathbf{w}, \mathbf{x}_i\rangle + b &\leq -1-\xi_i\,, & i\in\mathcal{I}_- \\ \xi_i &\geq 0\,, & i\in\mathcal{I}_+\cup\mathcal{I}_- \end{aligned}$$



- **Dual QP:** (m variables; $2m+1$ constraints)

$$\alpha^* = \underset{\alpha\in\mathbb{R}^m}{\arg\max} \left[\sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\,\alpha_j\,y_i\,y_j\,\langle \mathbf{x}_i, \mathbf{x}_j\rangle\right]$$

subject to

$$\sum_{i=1}^{m}\alpha_i\,y_i = 0\,, \qquad 0\leq\alpha_i\leq C\,, \quad i\in\mathcal{I}_-\cup\mathcal{I}_+$$

- The primal variables $\mathbf{w}^*, b^*$ can be computed from the dual variables α^*:

$$\mathbf{w}^* = \sum_{i=1}^{m}\alpha_i^*\,y_i\,\mathbf{x}_i$$

$$b^* = y_i - \langle \mathbf{w}^*, \mathbf{x}_i\rangle \quad \text{for} \quad \alpha_i^*\in(0, C)$$

Kernel Support Vector Machines

- The SVM dual formulation depends only on **dot products** $\langle \mathbf{x}, \mathbf{x}' \rangle \Rightarrow$ Replacing dot products by a **kernel function** $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ implicitly maps inputs into a feature space via

$$\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle = k(\mathbf{x}, \mathbf{x}')$$

where $\phi : \mathcal{X} \to \mathbb{R}^n$ is the (possibly infinite-dimensional) feature map.

- **Learning**: given training data $T_m = \big((\mathbf{x}_i, y_i) \in \mathcal{X} \times \{-1, +1\} \mid i = 1, \ldots, m\big)$ find:

$$\alpha^* = \arg\max_{\alpha \in \mathbb{R}^m} \left[\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \, \alpha_j \, y_i \, y_j \;\; k(\mathbf{x}_i \, , \mathbf{x}_j) \right]$$

subject to

$$\sum_{i=1}^{m} \alpha_i \, y_i = 0 \, , \qquad 0 \leq \alpha_i \leq C \, , \quad i \in \mathcal{I}_+ \cup \mathcal{I}_-$$

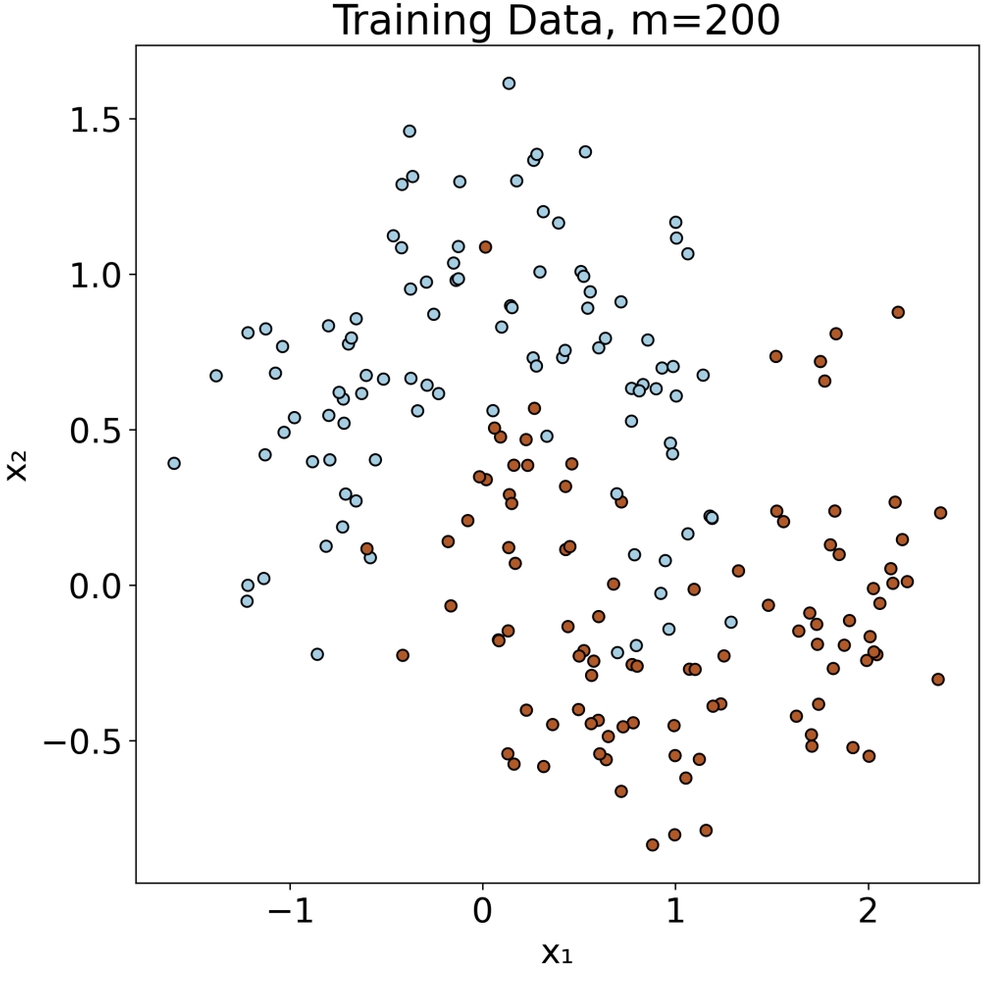
- **Prediction:**

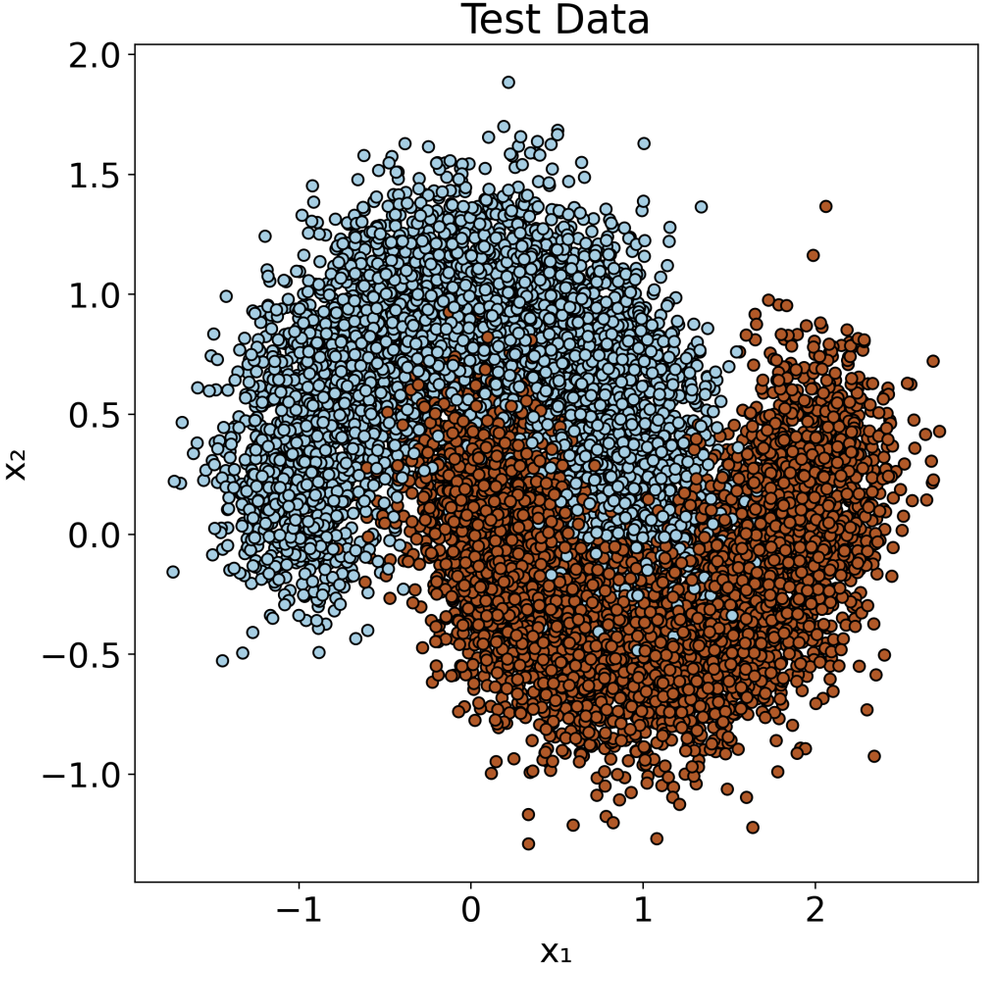
$$h(\mathbf{x}; \alpha, b) = \operatorname{sign}\Big(\langle \mathbf{w}^*, \phi(\mathbf{x}) \rangle + b^*\Big) = \operatorname{sign}\left(\sum_{i \in \mathcal{I}_{\mathrm{SV}}} \alpha_i^* \, y_i \;\; k(\mathbf{x}_i, \mathbf{x}) + b^* \right)$$

where $\mathcal{I}_{\mathrm{SV}} = \{i \in \{1, \ldots, m\} \mid \alpha_i^* > 0\}$ are indices of the support vectors – the only training samples that need to be stored for classification.

Example: Support Vector Machines with RBF kernel

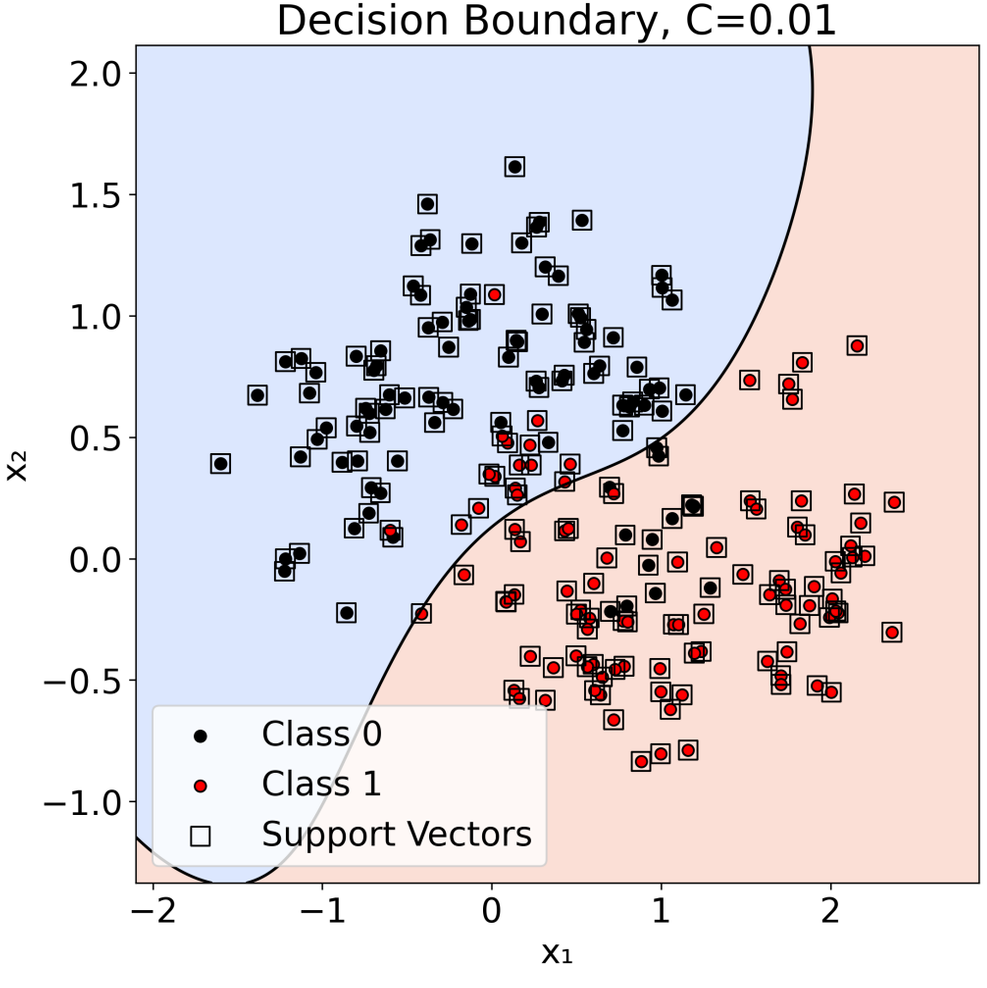
- RBF kernel: $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$ with $\gamma = 1$
- Input dimension $d = 2$; feature space dimension $n = \infty$
- Decision score: $f(\mathbf{x}) = \sum_{i \in \mathcal{I}_{\mathrm{SV}}} y_i \, \alpha_i^* \, \exp(-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2) + b^*$

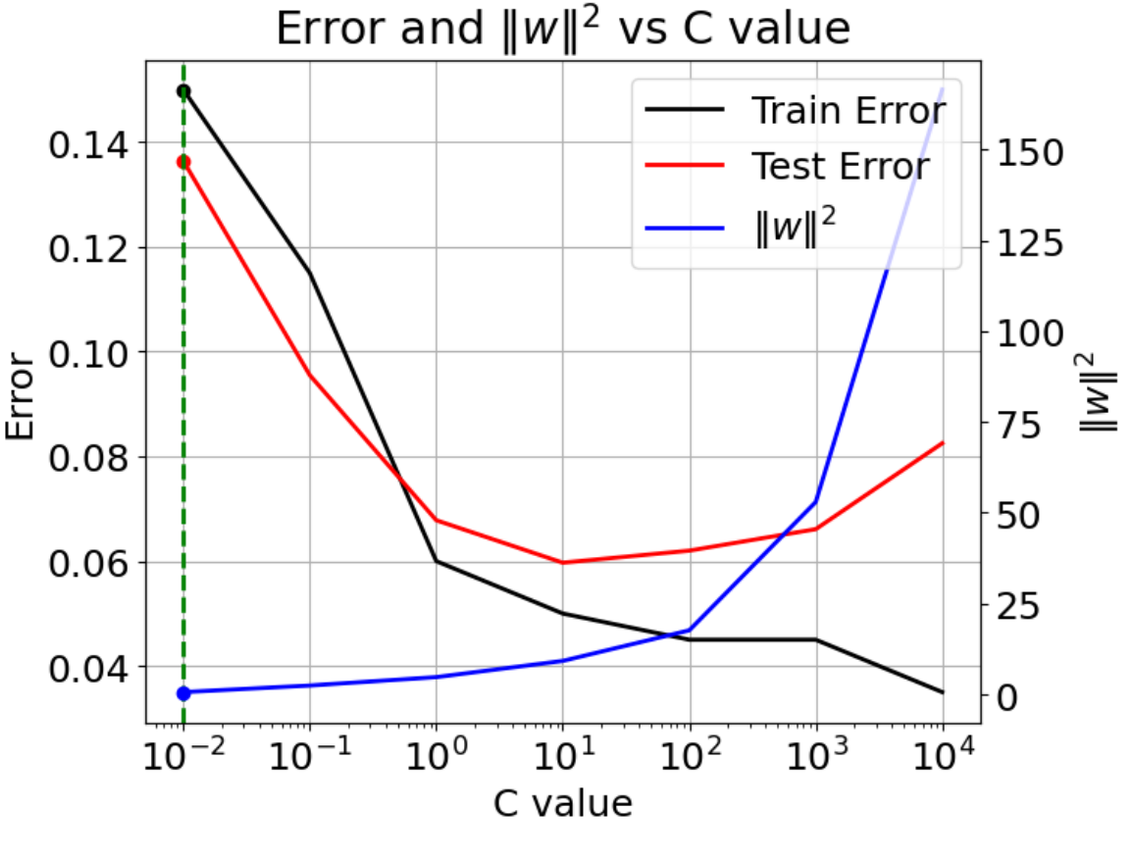




Example: Support Vector Machines with RBF kernel

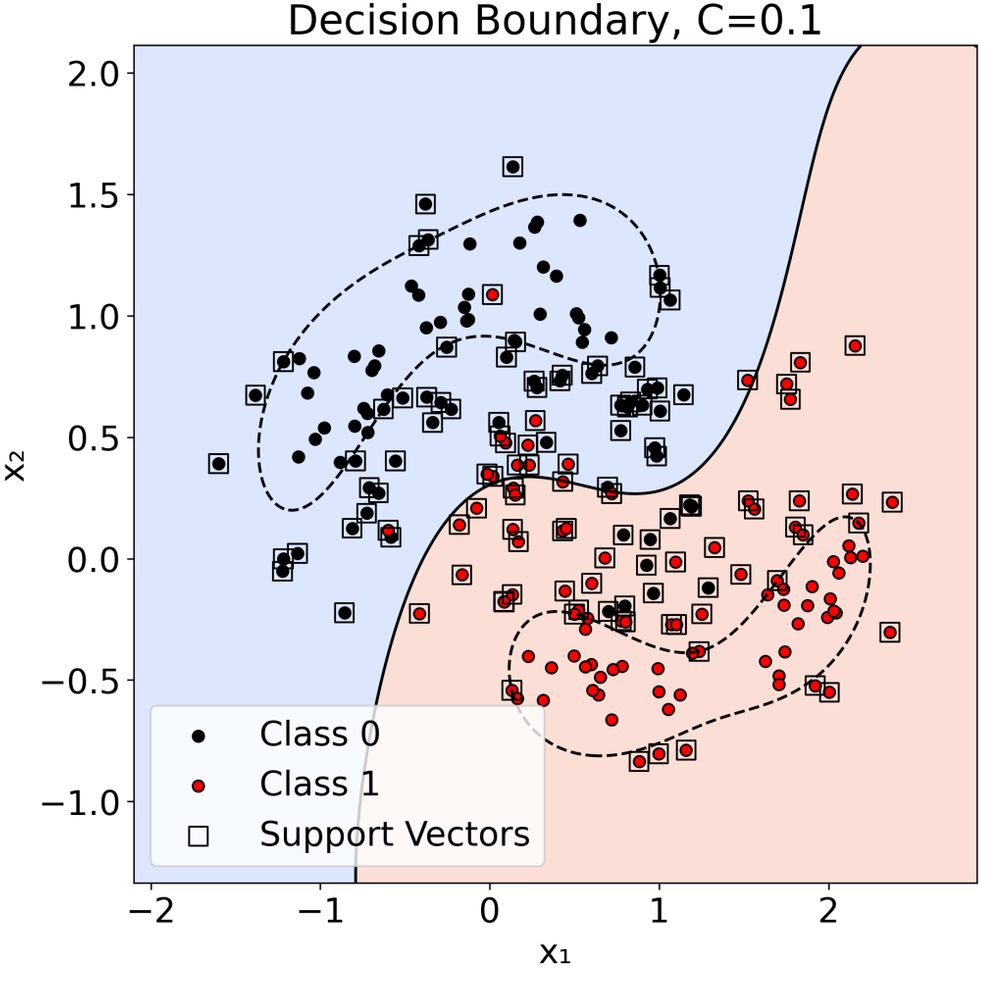
- RBF kernel: $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$ with $\gamma = 1$
- Input dimension $d = 2$; feature space dimension $n = \infty$
- Decision score: $f(\mathbf{x}) = \sum_{i \in \mathcal{I}_{\mathrm{SV}}} y_i \, \alpha_i^* \, \exp(-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2) + b^*$

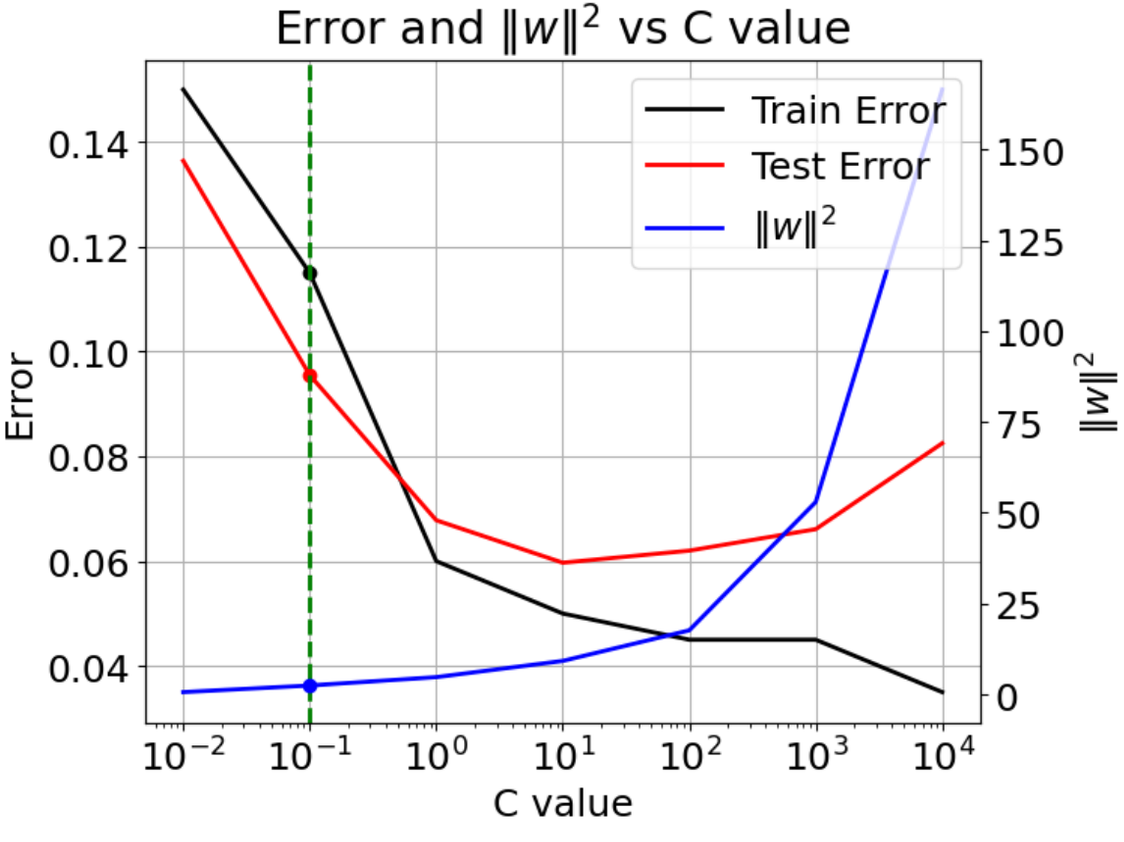




Example: Support Vector Machines with RBF kernel

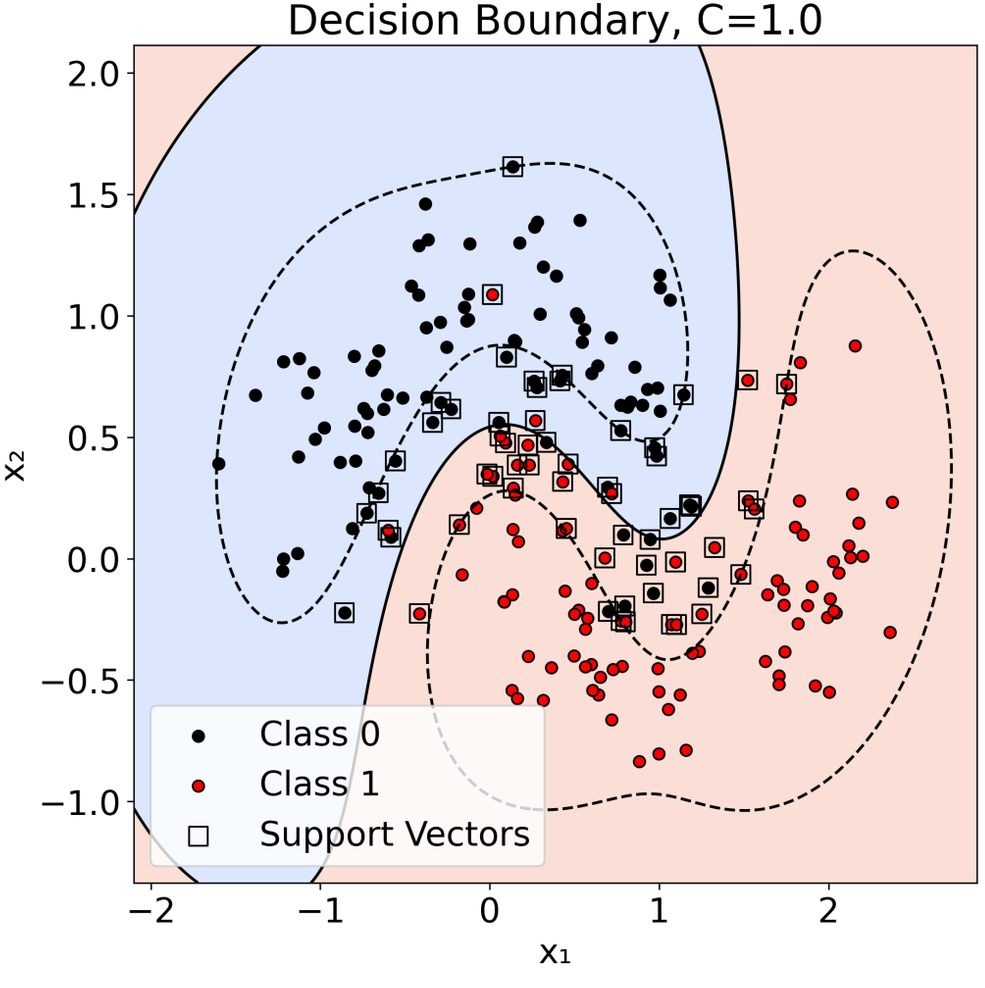
- RBF kernel: $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$ with $\gamma = 1$
- Input dimension $d = 2$; feature space dimension $n = \infty$
- Decision score: $f(\mathbf{x}) = \sum_{i \in \mathcal{I}_{\mathrm{SV}}} y_i \, \alpha_i^* \, \exp(-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2) + b^*$

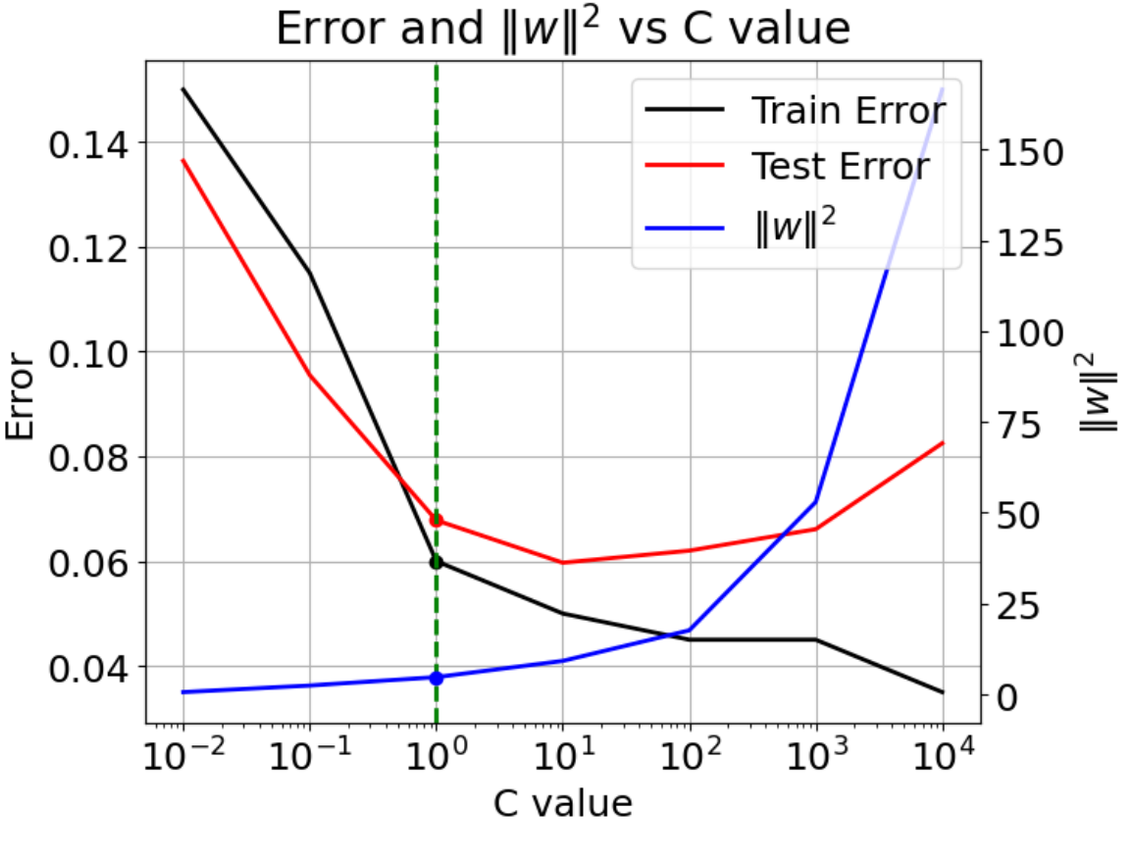




Example: Support Vector Machines with RBF kernel

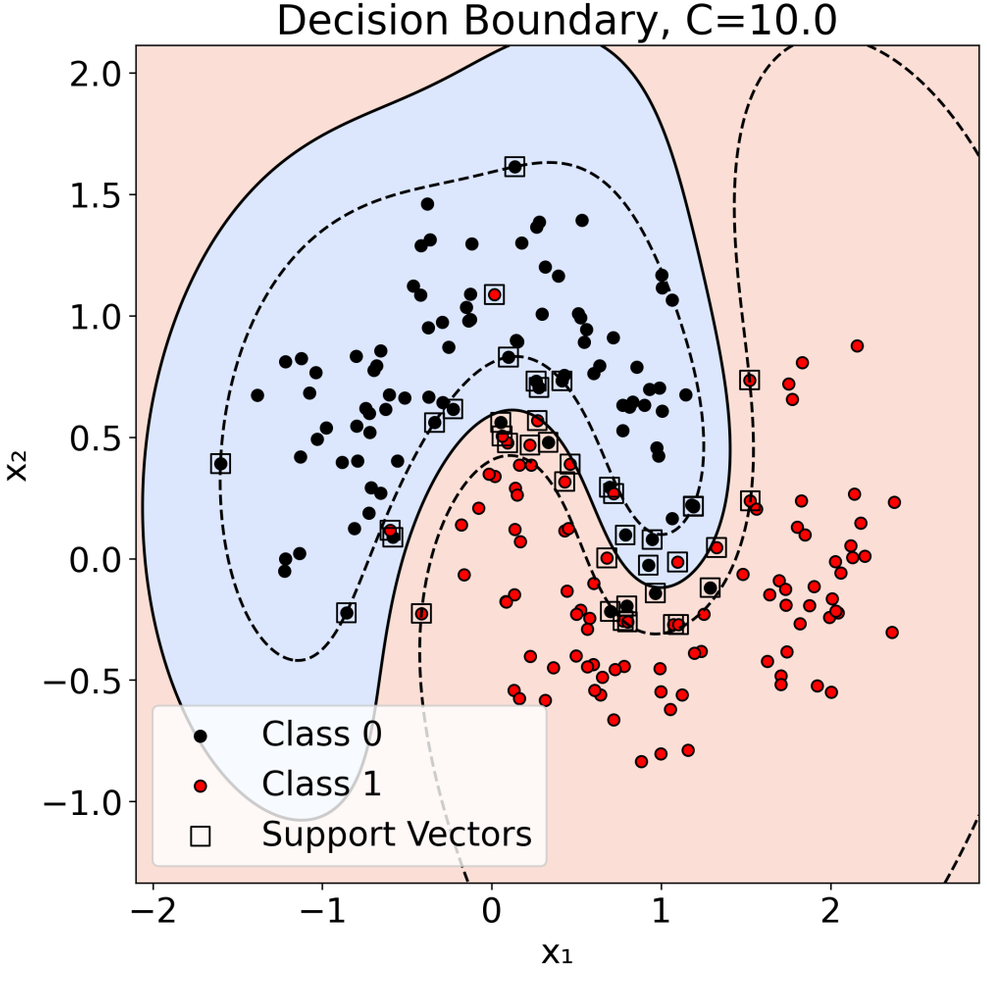
- RBF kernel: $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$ with $\gamma = 1$
- Input dimension $d = 2$; feature space dimension $n = \infty$
- Decision score: $f(\mathbf{x}) = \sum_{i \in \mathcal{I}_{\mathrm{SV}}} y_i \, \alpha_i^* \, \exp(-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2) + b^*$

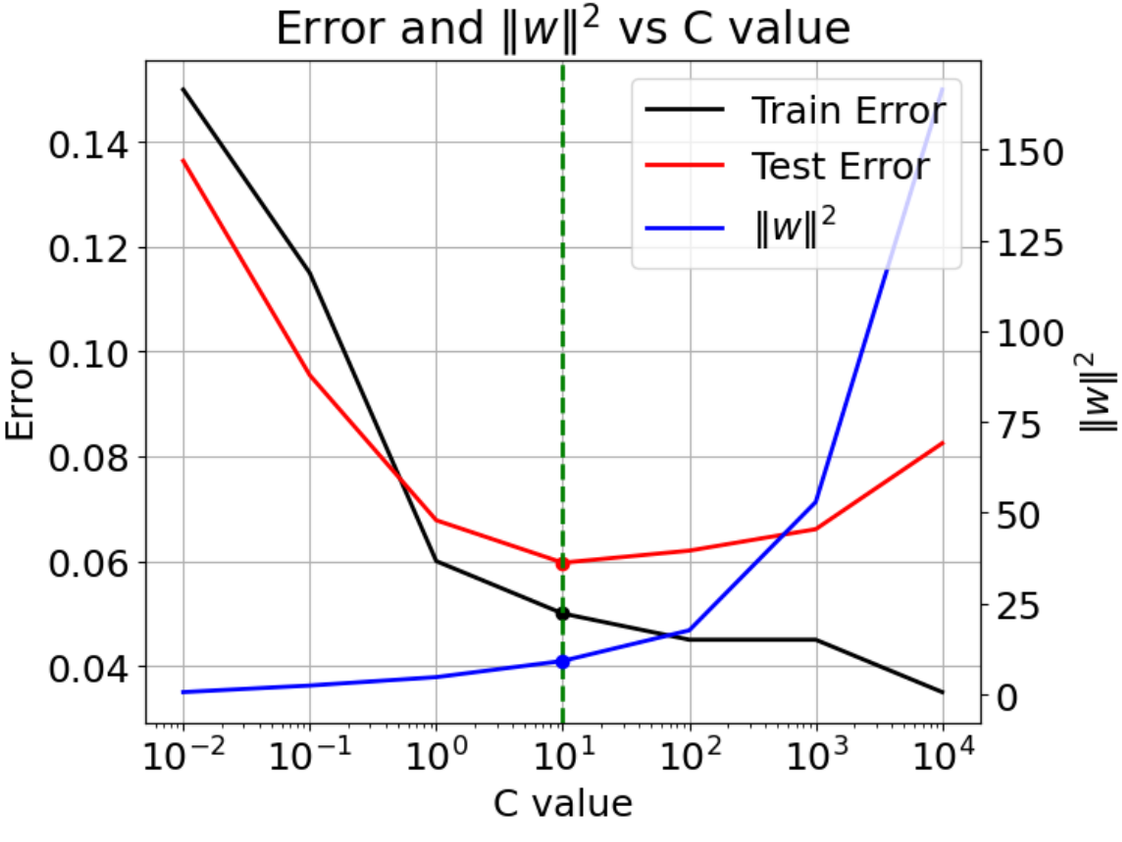




Example: Support Vector Machines with RBF kernel

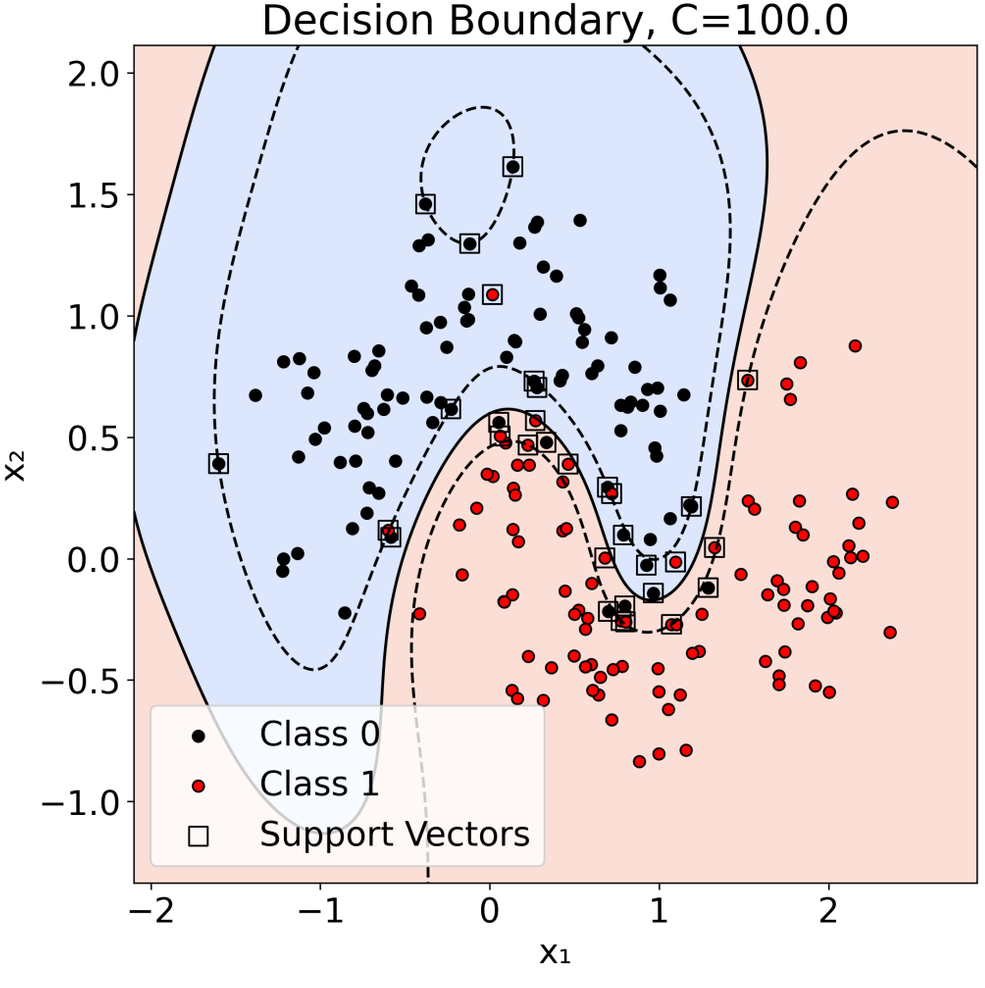
- RBF kernel: $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$ with $\gamma = 1$
- Input dimension $d = 2$; feature space dimension $n = \infty$
- Decision score: $f(\mathbf{x}) = \sum_{i \in \mathcal{I}_{\mathrm{SV}}} y_i \, \alpha_i^* \, \exp(-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2) + b^*$

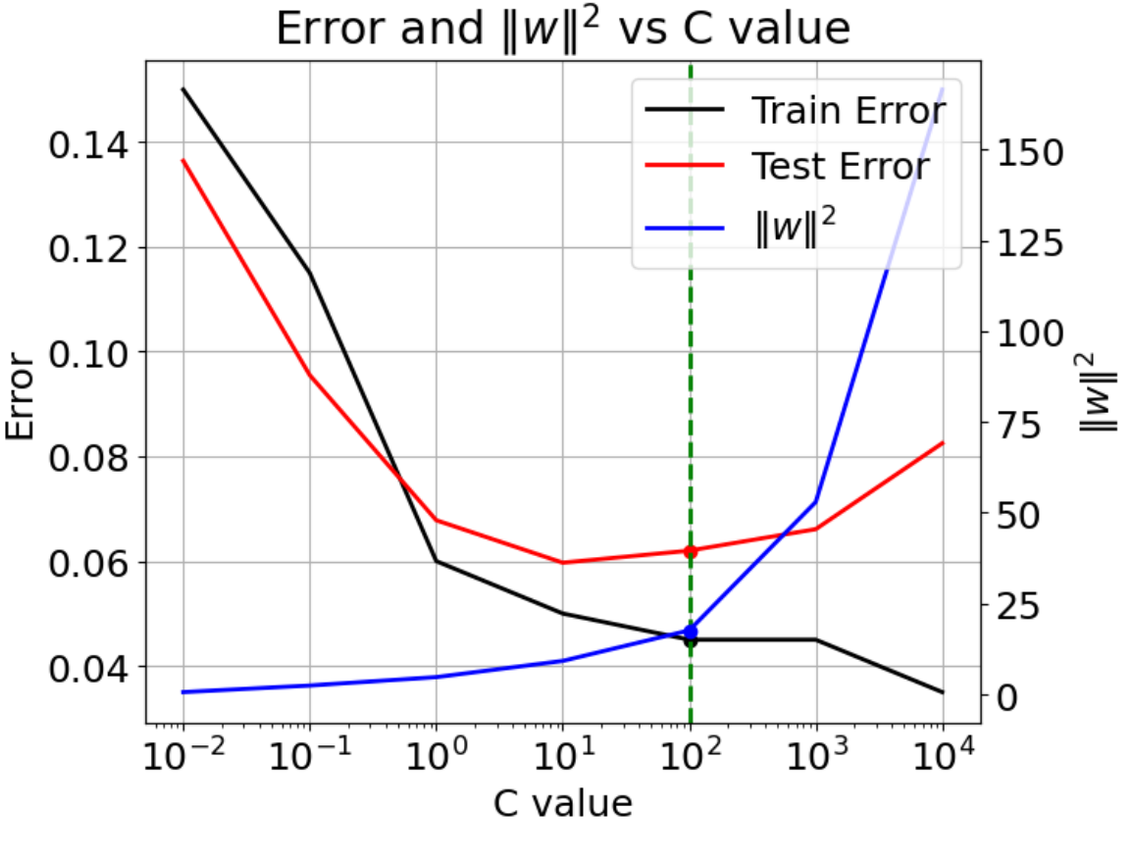




Example: Support Vector Machines with RBF kernel

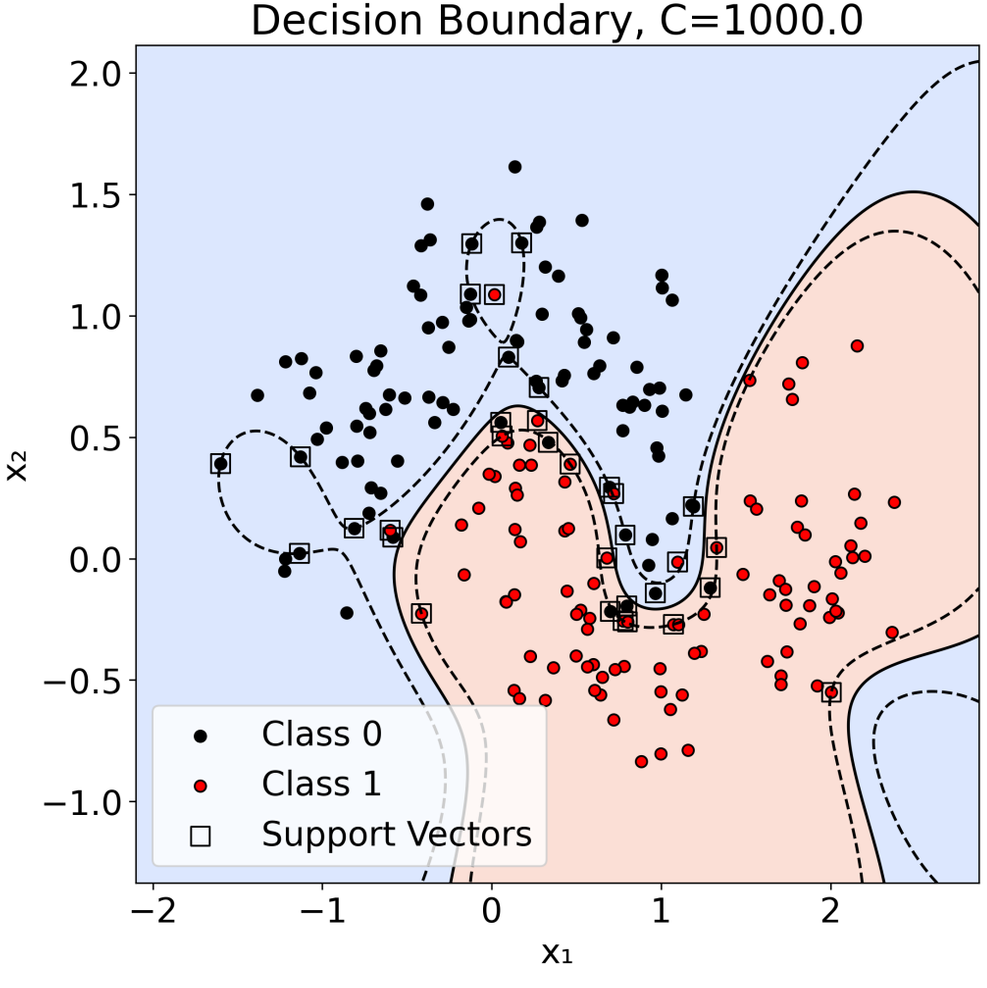
- RBF kernel: $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$ with $\gamma = 1$
- Input dimension $d = 2$; feature space dimension $n = \infty$
- Decision score: $f(\mathbf{x}) = \sum_{i \in \mathcal{I}_{\mathrm{SV}}} y_i \, \alpha_i^* \, \exp(-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2) + b^*$

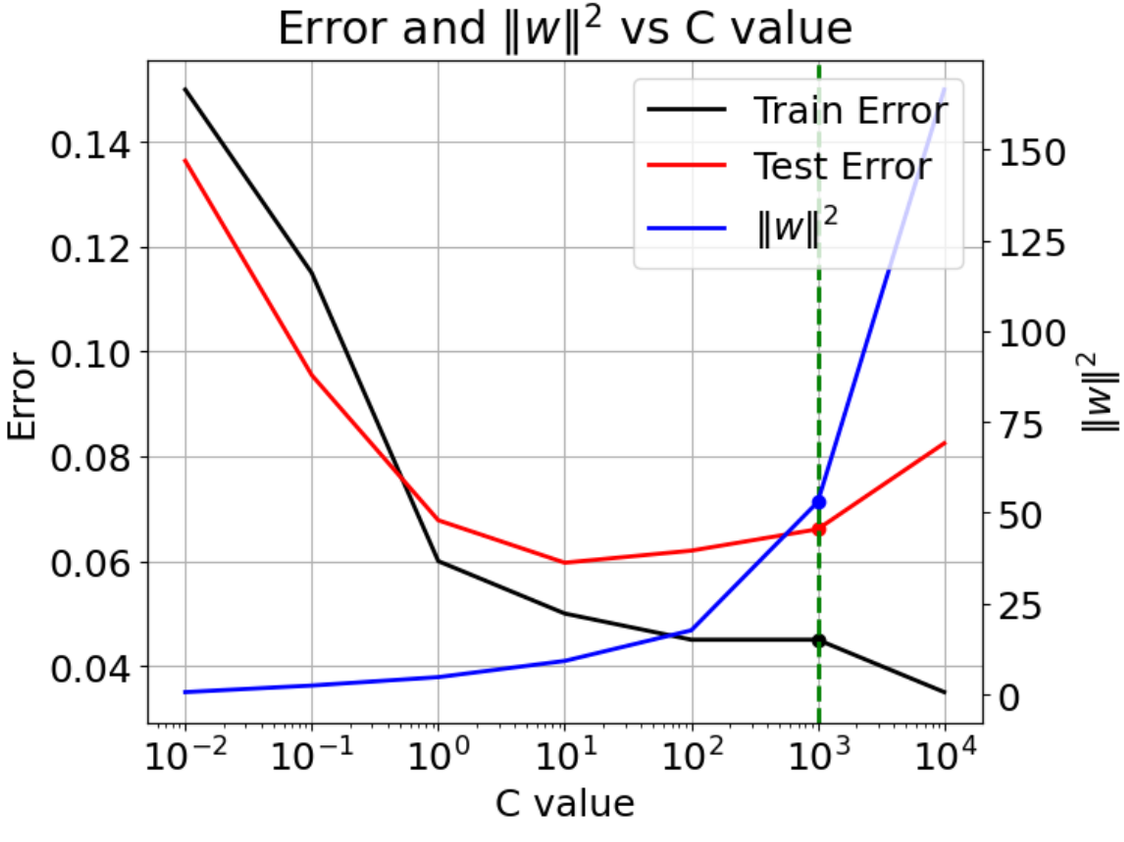




Example: Support Vector Machines with RBF kernel

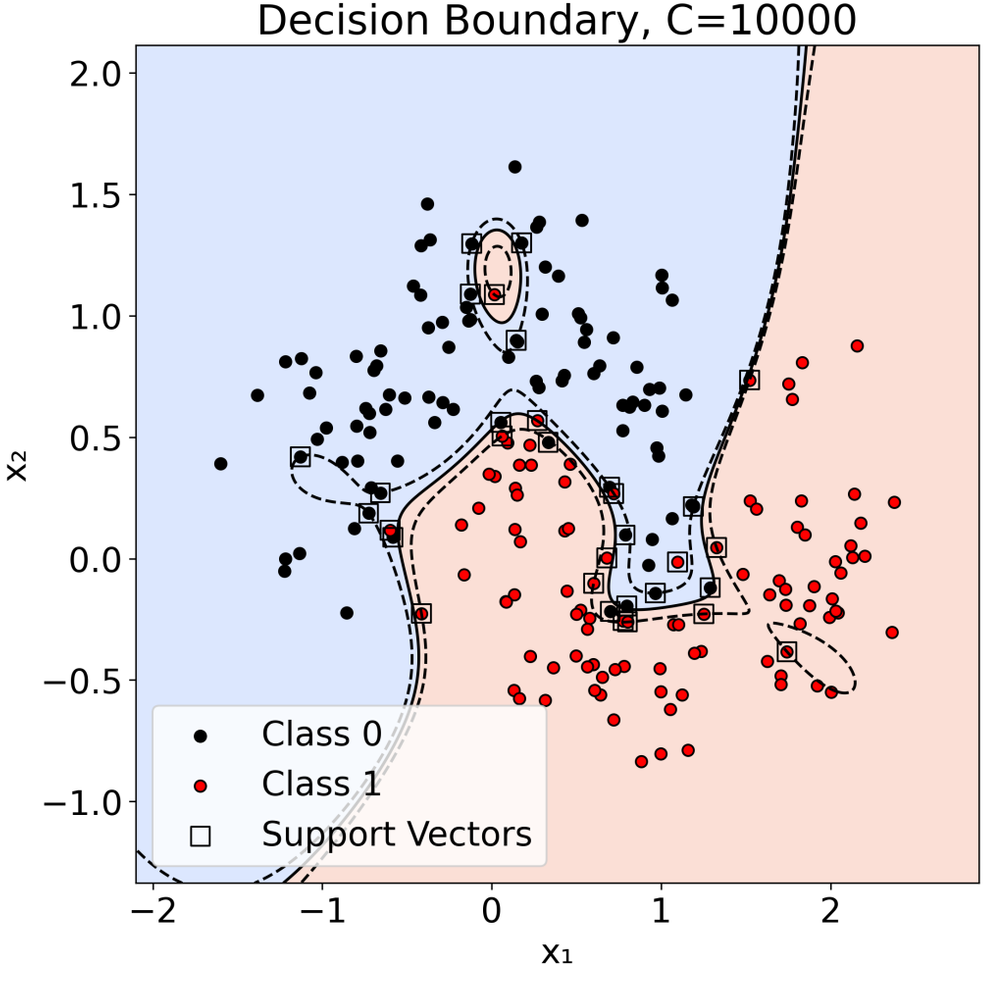
- RBF kernel: $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$ with $\gamma = 1$
- Input dimension $d = 2$; feature space dimension $n = \infty$
- Decision score: $f(\mathbf{x}) = \sum_{i \in \mathcal{I}_{\mathrm{SV}}} y_i \, \alpha_i^* \, \exp(-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2) + b^*$

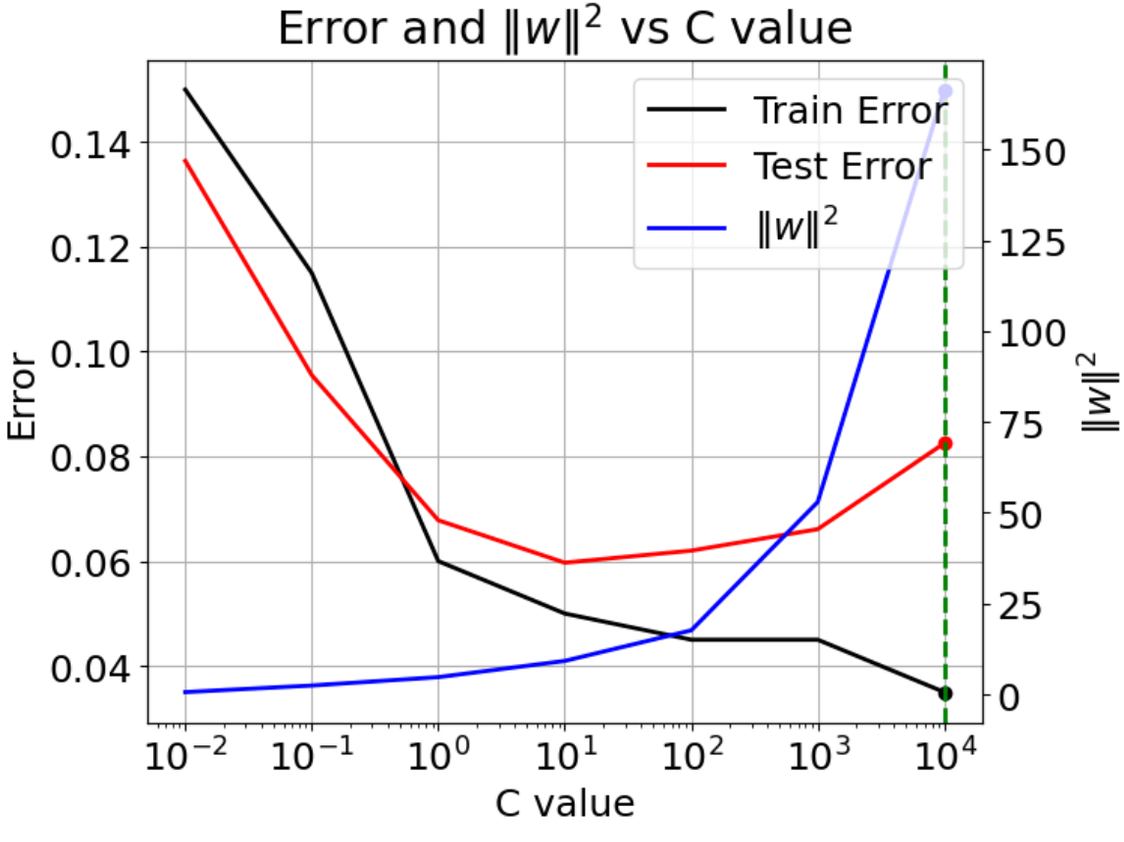




Example: Support Vector Machines with RBF kernel

- RBF kernel: $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$ with $\gamma = 1$
- Input dimension $d = 2$; feature space dimension $n = \infty$
- Decision score: $f(\mathbf{x}) = \sum_{i \in \mathcal{I}_{\mathrm{SV}}} y_i \, \alpha_i^* \, \exp(-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2) + b^*$





Linear vs. kernel SVM

Compute and memory requirements

	Training set memory	Prediction compute	Training complexity	When efficient?
Linear SVM	$\mathcal{O}(m \cdot n)$	$\mathcal{O}(n)$	$\mathcal{O}(n \cdot m)$–$\mathcal{O}(n \cdot m^2)$	large number of examples
Kernel SVM	$\mathcal{O}(m^2)$	$\mathcal{O}(\lvert\mathcal{I}_{\mathrm{SV}}\rvert)$	$\mathcal{O}(m^2)$–$\mathcal{O}(m^3)$	high-dimensional feature space

Examples of kernel functions

	$k(x, x')$	Input space $\mathcal{X}$	Feature space dimesnion n	Evaluation compute K
2nd polynomial	$\langle \mathbf{x}\,, \mathbf{x}' \rangle^2$	$\mathbb{R}^d$	$\frac{d(d+1)}{2}$	$\mathcal{O}(d)$
Degree p polynomial	$\langle \mathbf{x}\,, \mathbf{x}' \rangle^p$	$\mathbb{R}^d$	$\binom{n+p-1}{p}$	$\mathcal{O}(d)$
RBF	$\exp\left(-\gamma \lVert \mathbf{x} - \mathbf{x}' \rVert^2\right)$	$\mathbb{R}^d$	∞	$\mathcal{O}(d)$
String sub-sequence	dynamic programming	$\cup_{d=0}^{\infty} \Sigma^d$	$\lvert\Sigma\rvert^q$	$\mathcal{O}(q\lvert s\rvert\lvert t\rvert)$

m...number of training examples, d..input space dimension, n...feature space dimension, $|\mathcal{I}_{\mathrm{SV}}|$..number of support vectors

Summary: Support Vector Machines

- **Linear classification:** a classifier $h(\mathbf{x}; \mathbf{w}, b) = \mathrm{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$ separates the input space by a hyperplane.
- **Maximum-margin principle:** among all separating hyperplanes, SVM chooses the one with the largest margin, which improves robustness and generalization.
- **Hard-margin SVM:** for linearly separable data, delivers maximum margin linear classifier.
- **Soft-margin SVM:** for non-separable data, introduce slack variables $\xi_i \geq 0$, which is equivalent to minimizing regularized empirical risk with the **hinge loss**.
- **Support vectors:** only training examples on or inside the margin determine the final classifier.
- **Kernel SVM:** replacing dot products by a kernel $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$ allows linear separation in a feature space and nonlinear decision boundaries in the original space.
- **Practical takeaway:**
  - **Linear SVM:** efficient for large number of training examples.
  - **Kernel SVM:** more flexible, but training and memory are more expensive.
  - Hyperparameters such as C and kernel parameters (e.g. γ for RBF) control the trade-off between fitting and generalization.

Key idea: SVMs combine **large-margin learning**, **convex optimization**, and optionally the **kernel trick** to construct accurate linear or nonlinear classifiers.