Reliability assessment with SVR adaptive surrogate models

Jean-Marc Bourinet

SIGMA Clermont & Institut Pascal (CNRS UMR 6602)
Clermont-Ferrand, France

Email: bourinet@sigma-clermont.fr
FERUM open-source Matlab toolbox: http://www.ifma.fr/FERUM

Time-invariant reliability (a.k.a. static simulation problem)

- ✓ $\mathbf{X}$: **continuous random vector** ($\mathcal{X} \subseteq \mathbb{R}^n$) with known joint pdf $f_{\mathbf{X}}$
- ✓ g : limit-state function (LSF)
- ✓ $\mathcal{F}_{\mathbf{x}} = \{\mathbf{x} \in \mathcal{X} : g(\mathbf{x}) \leq 0\}$: failure domain

  $\mathcal{F}_{\mathbf{x}}^0 = \{\mathbf{x} \in \mathcal{X} : g(\mathbf{x}) = 0\}$: limit-state surface (LSS)
- ✓ p_{f} : **failure probability**

$$p_{\mathrm{f}} = \mathbb{P}(g(\mathbf{X}) \leq 0) = \int_{\mathcal{F}_{\mathbf{x}}} f_{\mathbf{X}}(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

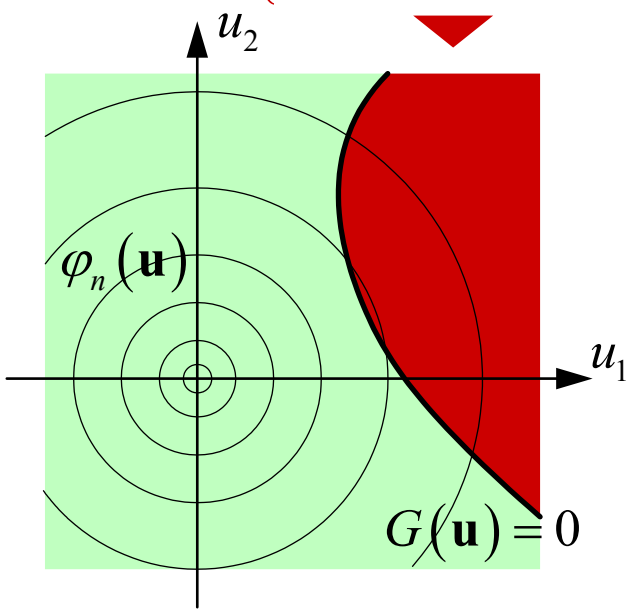
- ✓ Standard normal space formulation

$$\begin{aligned} p_{\mathrm{f}} &= \mathbb{P}(G(\mathbf{U}) \leq 0) \\ &= \int_{\mathcal{F}_{\mathbf{u}}} \varphi_n(\mathbf{u})\,\mathrm{d}\mathbf{u} \\ &= \mathbb{E}\left[1_{\mathcal{F}_{\mathbf{u}}}(\mathbf{U})\right] \end{aligned}$$

Rare event

$p_{\mathrm{f}} \ll 1$

$\mathcal{F}_{\mathbf{u}} = \{\mathbf{u} \in \mathbb{R}^n : G(\mathbf{u}) \leq 0\}$



Surrogate models

Learning framework

Input parameter space: $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ space of data

Output parameter space: $\mathbf{y} \in \mathcal{Y}$ space of responses

Scope of the presentation:

- ✓ **Scalar** output y
- ✓ Discrete set $\mathcal{Y}$, e.g. $\mathcal{Y} = \{-1,\ +1\}$
  ▶ **Classification** (e.g. pattern recognition)
- ✓ Continuous set $\mathcal{Y}$: $\mathcal{Y} = \mathbb{R}$
  ▶ **Regression**

Input data

N data pairs: $\{ (\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y},\ i = 1, \ldots, N \}$ ▶ **Training set** $\mathcal{T}$

Objective

Construct a model $\tilde{f}(\mathbf{x})$ which predicts the output at any **new** $\mathbf{x} \in \mathcal{X}$

$\tilde{f} : \mathcal{X} \to \mathcal{Y}$ $\quad \mathbf{x} \mapsto \tilde{y} = \tilde{f}(\mathbf{x})$ **Regression**

$\mathbf{x} \mapsto \tilde{y} = \operatorname{sign} \tilde{f}(\mathbf{x})$ **Binary classification**

Optimal approximate model

Construction of an "optimal" approximate model

The model is expected to predict well not only over the training data but also (and more importantly) **over unseen data**.

▶ **Ability of the model to generalize well**

$\tilde{f}$ chosen from a known set of candidate functions (hypothesis space $\mathcal{H}$ for SVMs) by minimization of an error criterion defined from the training set (our **only** source of information)

How (im)perfect is this "optimal" model?

The approximate model is chosen from a **set of specified models**
- for polynomial approximations: degree, basis
- for kriging: assumption about trend, type of covariance function, …
- for SVMs: type of loss function, type of kernel, type of regularization

We do not know if the selected type of approximate model is sufficiently "flexible" to obtain a good approximation of the true model!

Surrogate of a numerical model

Where do y-data come from?

We are given a physical model (referred to as **true model**):

$$f: \quad \mathcal{X} \to \mathcal{Y}$$
$$\mathbf{x} \mapsto y = f(\mathbf{x})$$

The training set now becomes: $\mathcal{T} = \left\{ \left(\mathbf{x}_i, f(\mathbf{x}_i)\right),\ i = 1, \ldots, N \right\}$

The true model is often costly to evaluate (e.g. a FE model).
► **We want to construct an approximation of f from a training set $\mathcal{T}$ as small as possible (small N)**

$\widetilde{f}$ is called **surrogate model** (**machine** in statistical learning, **metamodel** or **response surface** in engineering, …)

Application contexts of surrogate models

Deterministic optimization

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \quad \text{subject to} \begin{cases} g_i(\mathbf{x}) \leq 0 \quad , \quad i = 1,\ldots,n_g \\ h_j(\mathbf{x}) = 0 \quad , \quad j = 1,\ldots,n_h \end{cases}$$

Global sensitivity analysis (e.g. by means of Sobol' indices)

$$S_i = \frac{V_i}{\mathrm{Var}(Y)} = \frac{\mathrm{Var}\left(\mathbb{E}[Y \mid X_i]\right)}{\mathrm{Var}(Y)} \quad , \quad S_{ij} = \frac{V_{ij}}{\mathrm{Var}(Y)} = \frac{\mathrm{Var}\left(\mathbb{E}[Y \mid X_i, X_j]\right) - V_i - V_j}{\mathrm{Var}(Y)}$$

where $Y = f(\mathbf{X})$

Reliability assessment

$$p_{\mathrm{f}} = \mathbb{P}\left(g(\mathbf{X}) \leq 0\right) = \int_{\mathcal{F}_x} f_{\mathbf{X}}(\mathbf{x})\,\mathrm{d}\,\mathbf{x} = \mathbb{E}_{f_{\mathbf{X}}}\left[1_{\mathcal{F}_x}(\mathbf{X})\right]$$

$$= \mathbb{P}\left(G(\mathbf{U}) \leq 0\right) = \int_{\mathcal{F}_{\mathbf{u}}} \varphi_n(\mathbf{u})\,\mathrm{d}\,\mathbf{u} = \mathbb{E}_{\varphi_n}\left[1_{\mathcal{F}_{\mathbf{u}}}(\mathbf{U})\right]$$

Main types of surrogate models

Most usual types of surrogate models

- ✓ Polynomial response surfaces, polynomial moving least squares
- ✓ Polynomial chaos expansions (PCE)
- ✓ Kriging (a.k.a. Gaussian prediction)
- ✓ **Support vector machines (SVM)**
- ✓ Artificial neural networks (ANN), radial basis function (RBF)

A few others

- ✓ High dimensional representation method (HDMR)
- ✓ Multivariate adaptive regression splines (MARS)

Adaptive surrogates

Accuracy of surrogate models

Unnecessary to construct surrogates highly accurate everywhere in $\mathcal{X}$

- ✓ accuracy close to the LSS for reliability assessment,
- ✓ accuracy close to the optimum in optimization.

From passive to active learning

- ✓ Passive strategy:
  - Space filling designs
  - Same level of accuracy sought for everywhere in $\mathcal{X}$
- ✓ Adaptive strategy (a.k.a. active learning):
  - Initial training set, enriched by additional training pairs
  - Selection of new training pairs based on the information conveyed by constructed surrogate models
  - Accuracy focused on important zones of $\mathcal{X}$

Adaptive surrogates – Reliability assessment

Problem setup

For $s = 1, \ldots, s_{\text{final}}$, we define:

- ✓ **surrogate models** $\widetilde{f}_s$
- ✓ additional data pairs $\mathcal{D}_s = \{ (\mathbf{x}_j, y_j), j = 1, \ldots, N_s \}$
  where N_s is the number of additional data pairs at iteration s
- ✓ training set $\mathcal{T}_s$, e.g. $\mathcal{T}_s = \bigcup_{k=1}^{s} \mathcal{D}_s$
- ✓ failure probability estimates: $\widehat{p}_s$

Objective

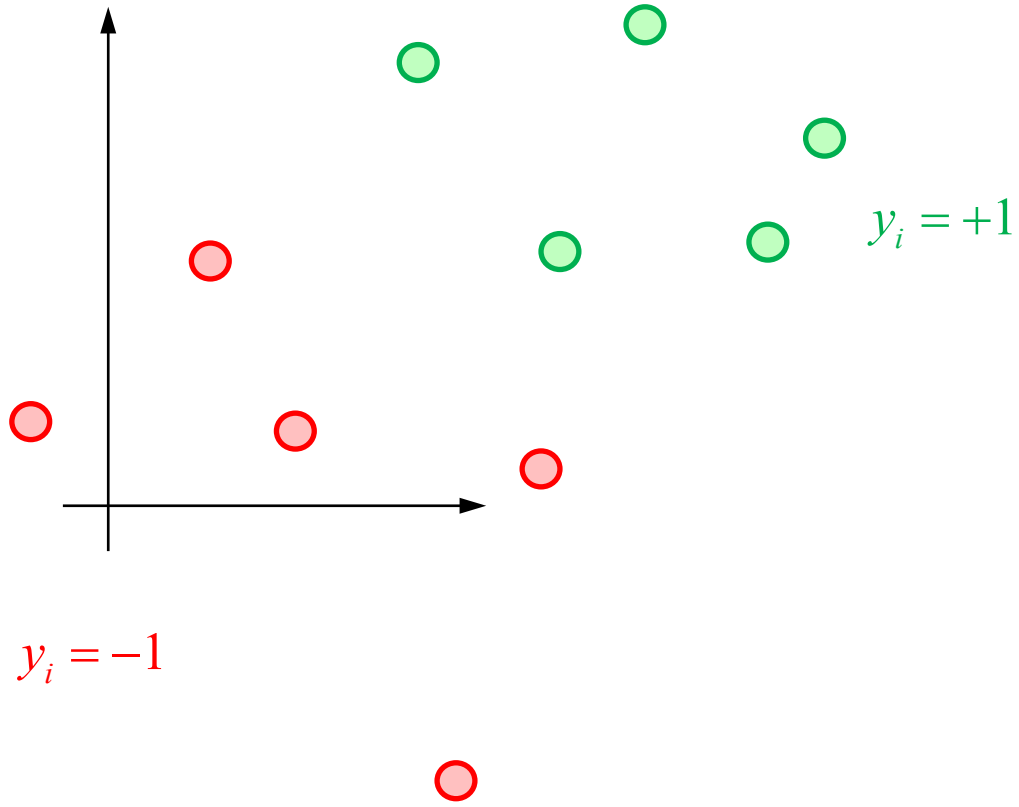
$$\min N_t = \sum_{s=1}^{s_{\text{final}}} N_s \quad \text{subject to} \quad \varepsilon\left(\widehat{p}_{s_{\text{final}}}, p_{\text{f ref}} \right) \leq \varepsilon_{\text{tol}}$$

- ✓ Optimization carried out over all the strategies that can be devised, i.e. numbers of new data pairs N_s and locations of points in $\mathcal{X}$
- ✓ A preference may be given for more than one point added at a time (distributed LSF evaluations on multi-core systems)

Linear classification with Support Vector Machines (SVC)

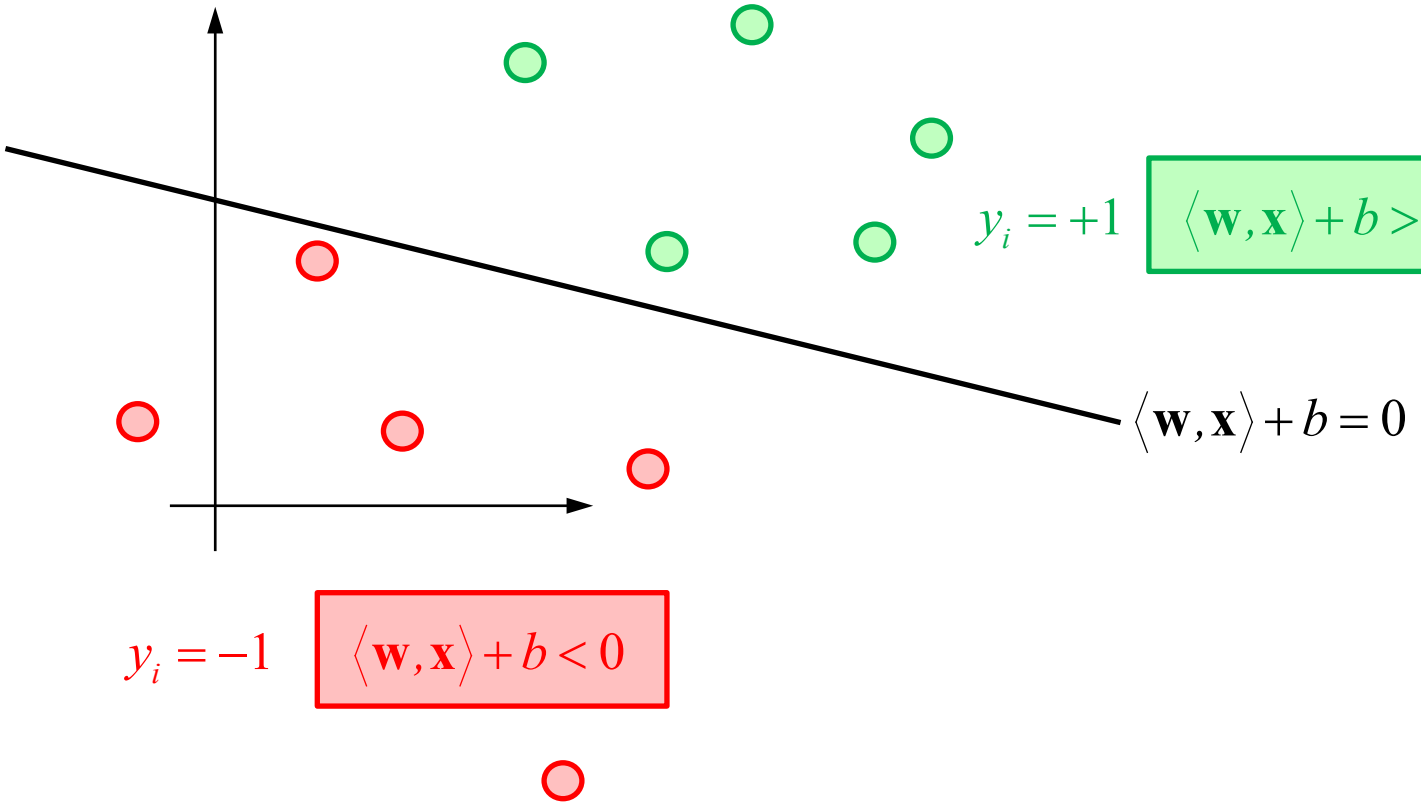
1 – Hard margin

SVM classification – Linearly separable data



N data pairs: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \in \mathbb{R}^n \times \{-1, 1\}$ ▶ **Training set**

Decision function

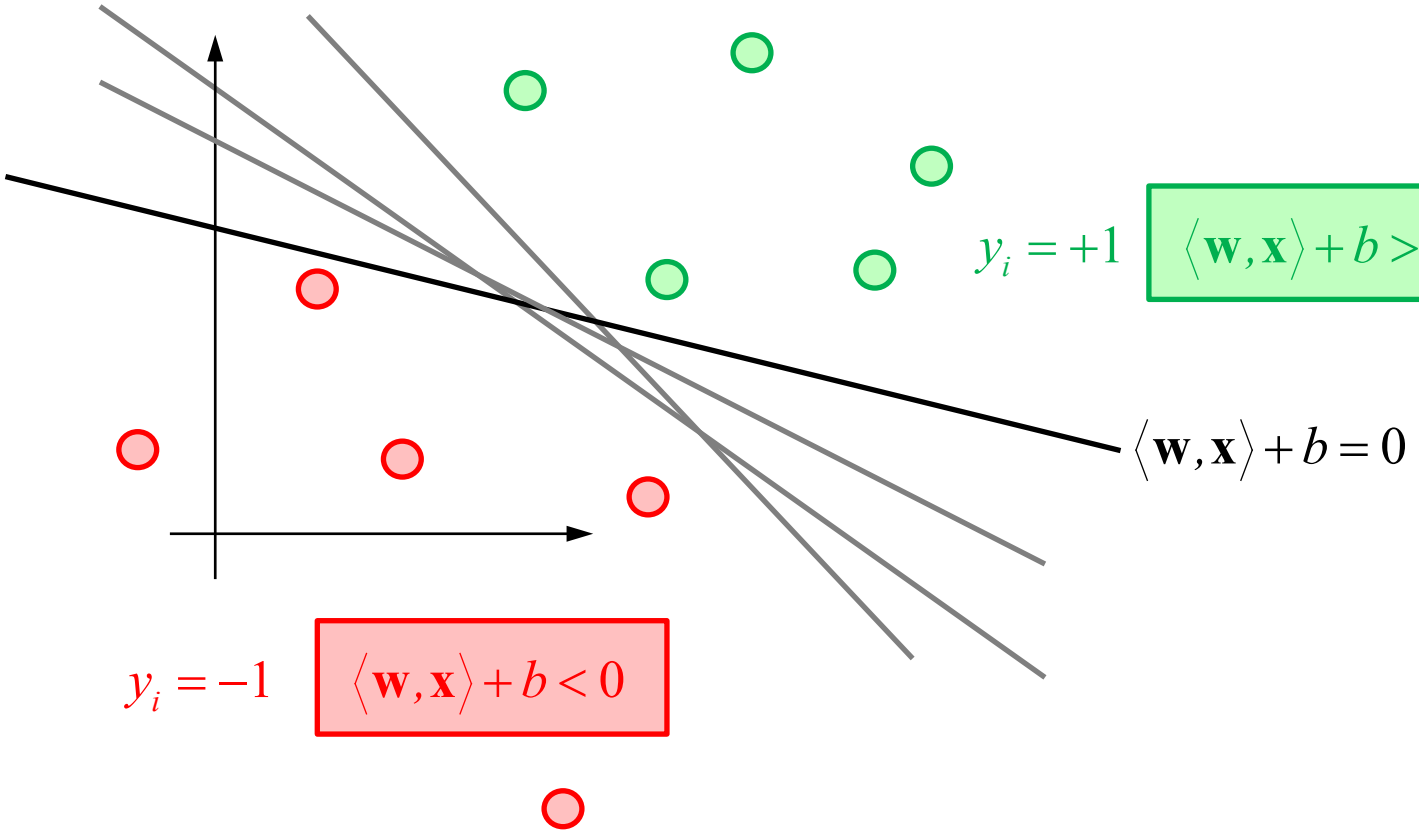


Linear **decision function**: $\tilde{y} = \operatorname{sign} \tilde{f}(\mathbf{x})$

where: $\tilde{f}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$

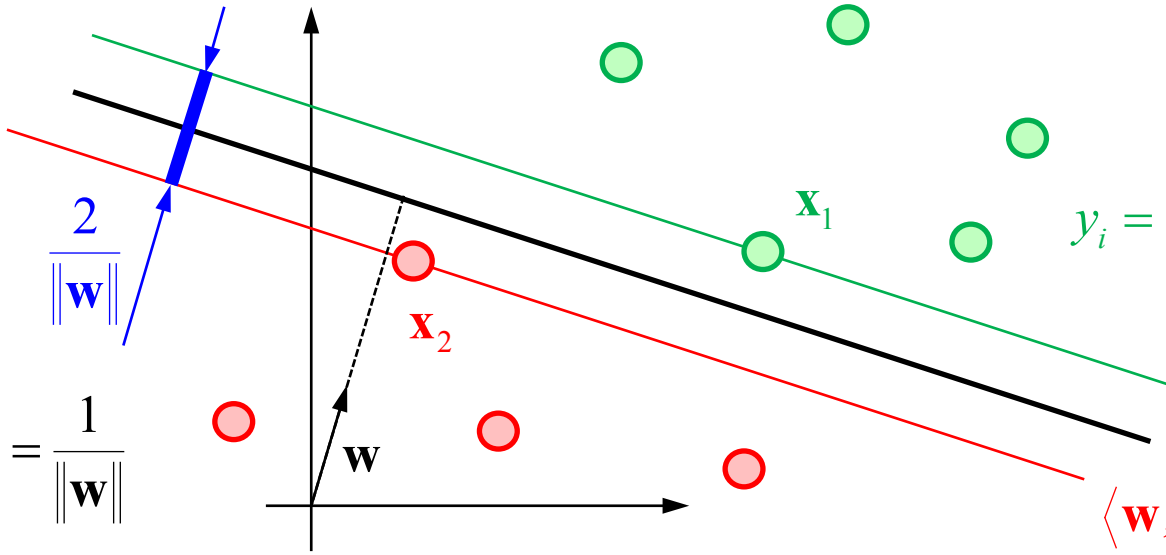
and $\mathbf{w} \in \mathbb{R}^n$ (**weight vector**), $b \in \mathbb{R}$ (**bias term**)

Several linear classifiers as candidates



Several possible solutions!

Canonical form, margin



Margin $= \frac{1}{\|\mathbf{w}\|}$

Canonical form
(by rescaling $\mathbf{w}$ and b)

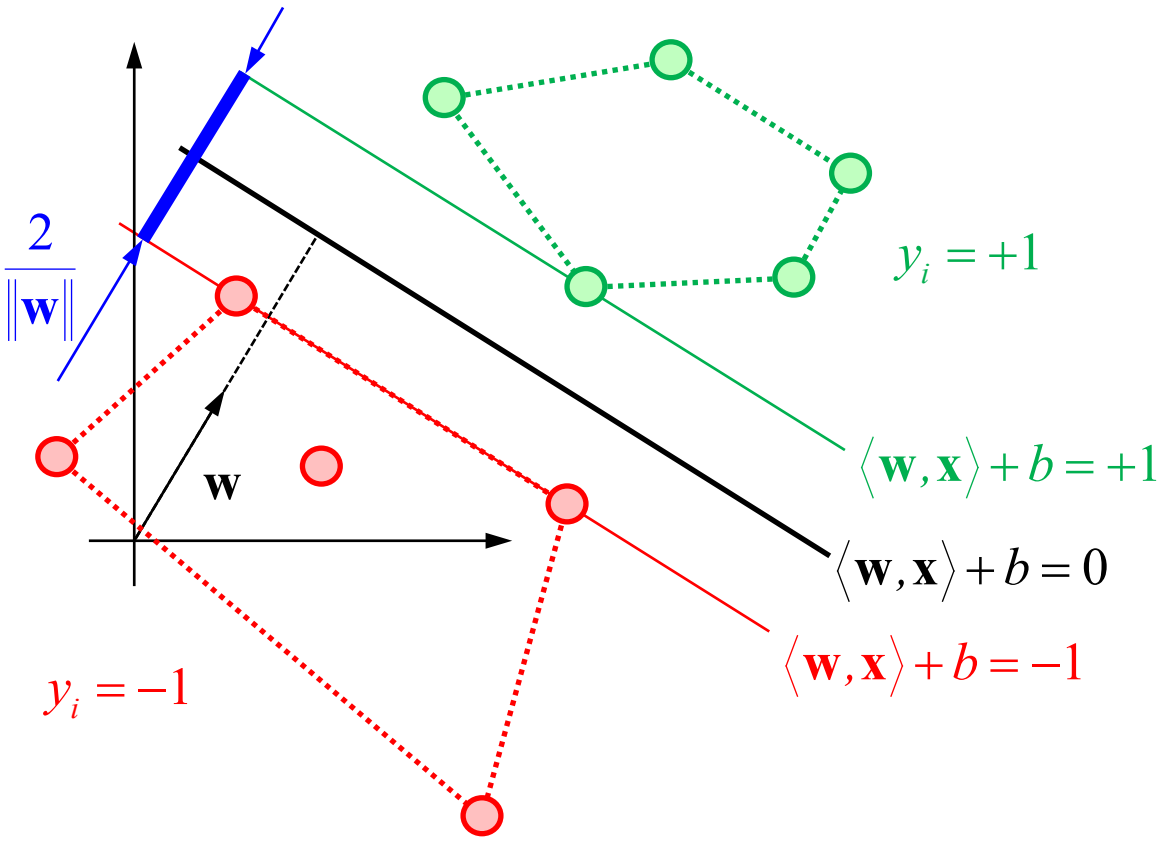
Canonical form of the decision function if: $\min_i \left| \langle \mathbf{w}, \mathbf{x}_i \rangle + b \right| = 1$

$\langle \mathbf{w}, \mathbf{x}_1 \rangle + b = +1$

$\langle \mathbf{w}, \mathbf{x}_2 \rangle + b = -1$

$\blacktriangleright \quad \langle \mathbf{w}, (\mathbf{x}_1 - \mathbf{x}_2) \rangle = 2 \quad \blacktriangleright \quad \left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, (\mathbf{x}_1 - \mathbf{x}_2) \right\rangle = \frac{2}{\|\mathbf{w}\|}$

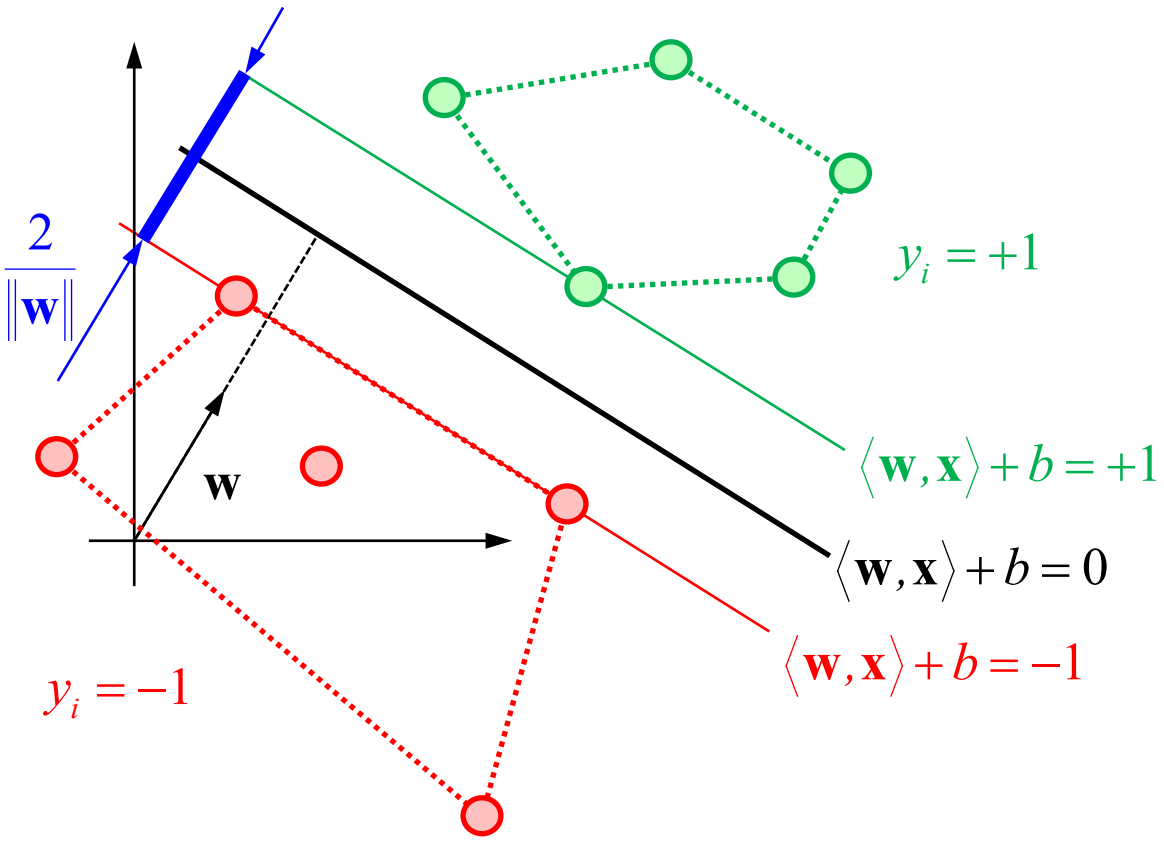
Optimal classifier



The **optimal** hyperplane classifier:

- ✓ is orthogonal to the shortest line connecting the convex envelope of the two classes (red and green dotted lines),
- ✓ intersects it half-way between the two classes.

Optimization problem



Formulation of the optimization problem to be solved:

$$\max_{\mathbf{w},b} \frac{2}{\|\mathbf{w}\|} \quad \text{subject to} \quad \begin{cases} \langle \mathbf{w}, \mathbf{x}_i \rangle + b \geq 1 & \text{if } y_i = +1 \quad \text{for } i = 1, \ldots N \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b \leq -1 & \text{if } y_i = -1 \quad \text{for } i = 1, \ldots N \end{cases}$$

Constraints for a correct classification of data

Optimization problem (primal)

Optimization problem to be solved

$$\max_{\mathbf{w},b} \frac{2}{\|\mathbf{w}\|} \quad \text{subject to} \quad \begin{cases} \langle \mathbf{w}, \mathbf{x}_i \rangle + b \geq 1 & \text{if } y_i = +1 \quad \text{for } i = 1,\ldots,N \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b \leq -1 & \text{if } y_i = -1 \quad \text{for } i = 1,\ldots,N \end{cases}$$

Equivalent optimization problem to be solved (primal form)

Maximizing the margin is equivalent to minimizing the norm of the weight vector (or here its squared norm)

$$\min_{\mathbf{w},b} \frac{\|\mathbf{w}\|^2}{2} \quad \text{subject to} \quad y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) \geq 1 \quad \text{for} \quad i = 1,\ldots,N$$

The optimization problem is a **convex and quadratic program**

Unique global minimum (when feasible)

Convex program = Constrained optimization problem + Convex objective function + Convex inequality constraint functions + Affine equality constraint functions

Quadratic program = Constrained optimization problem + Quadratic objective function + Affine (equality or inequality) constraint functions

From primal to dual form (Lagrangian)

Lagrangian (which is here differentiable)

$$L(\mathbf{w},b,\boldsymbol{\alpha}) = \frac{\|\mathbf{w}\|^2}{2} - \sum_{i=1}^{N} \alpha_i \left[y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) - 1 \right]$$

Introduction of **Lagrange multipliers** $\alpha_i \geq 0$

$L(\mathbf{w},b,\boldsymbol{\alpha})$ minimized w.r.t. primal variables $\mathbf{w}$ and b,
and maximized w.r.t. dual variables α_i (saddle point solution)

Karush-Kuhn-Tucker conditions

$$\frac{\partial L(\mathbf{w},b,\boldsymbol{\alpha})}{\partial b} = 0 \quad \blacktriangleright \quad \sum_{i=1}^{N} \alpha_i y_i = 0$$

$$\frac{\partial L(\mathbf{w},b,\boldsymbol{\alpha})}{\partial \mathbf{w}} = 0 \quad \blacktriangleright \quad \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$$

Lagrangian (rewritten)

$$L(\mathbf{w},b,\boldsymbol{\alpha}) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \sum_{i=1}^{N} \alpha_i$$

Optimization problem (Wolfe dual)

Wolfe dual of the optimization problem

$$\max_{\boldsymbol{\alpha}} \; -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle + \sum_{i=1}^{N}\alpha_i$$

$$\text{subject to} \quad \begin{cases} \sum_{i=1}^{N}\alpha_i y_i = 0 \\ \alpha_i \geq 0 \quad \text{for} \quad i = 1,\ldots,N \end{cases}$$

Important note: The constraint $\sum_{i=1}^{N}\alpha_i y_i = 0$ results from the presence of the bias term b.

Support vectors, bias term

KKT complementary slackness condition

$$\alpha_i \left[y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) - 1 \right] = 0 \quad \text{for} \quad i = 1, \ldots, N$$

Case 1: $y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) \neq 1 \quad \blacktriangleright \quad \alpha_i = 0$

$\mathbf{x}_i$ is irrelevant for the construction of the decision function

Case 2: $\alpha_i \neq 0 \quad \blacktriangleright \quad y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) = 1$

$\mathbf{x}_i$ is a **support vector**

Notation: $\mathcal{I}_{\text{sv}} = \{ i : \alpha_i \neq 0 \}$

Bias term

For any $i \in \mathcal{I}_{\text{sv}}$, we have: $y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) = 1$

$$\blacktriangleright \quad b = y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle \quad \text{where} \quad \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$$

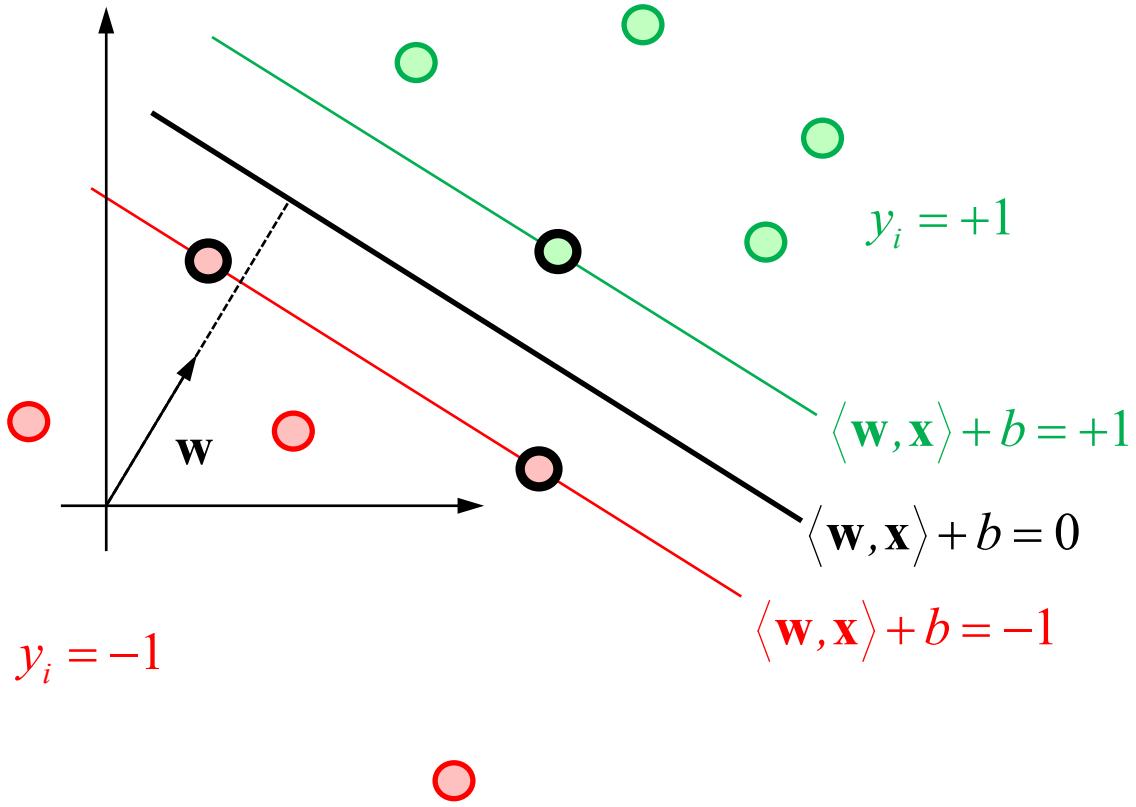
Recommended value for numerical stability: $b = \frac{1}{|\mathcal{I}_{\text{sv}}|} \sum_{i \in \mathcal{I}_{\text{sv}}} \left(y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle \right)$

Linear classification with Support Vector Machines (SVC)

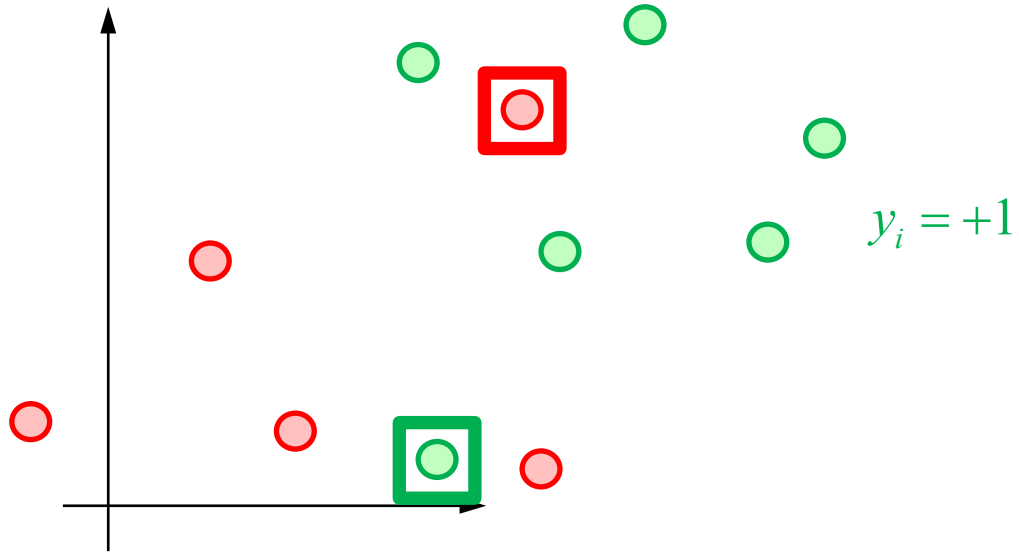
1 – Soft margin

Hard margin classifier (summary)

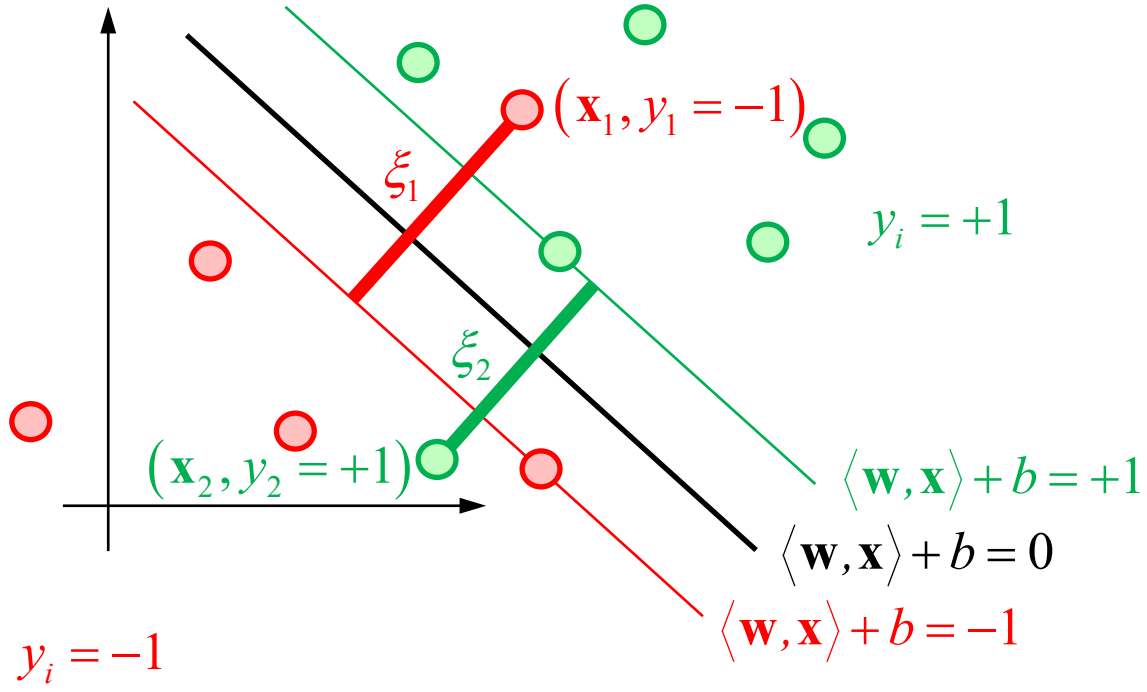
: **support vectors** ($\alpha_i \neq 0$)



From separable to nonseparable (noisy) data



Soft margin classification, regularization, slack variables



$$\min_{\mathbf{w},b,\xi} \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{N} \xi_i \quad \text{subject to} \quad \begin{cases} y_i\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \quad \text{for } i = 1, \ldots, N$$

ξ_i : **slack variables** (margin constraints allowed to be violated)

C : **regularization parameter** (violations of constraints are penalized)

Note: The above described optimization problem is specific to **L1-SVC**

Soft margin classification, primal optimization problem

Primal optimization problem (L1-SVC)

$$\min_{\mathbf{w},b,\xi} \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{N}\xi_i \quad \text{subject to} \quad \begin{cases} y_i\left(\langle \mathbf{w},\mathbf{x}_i\rangle + b\right) \geq 1-\xi_i & \text{for } i=1,\ldots,N \\ \xi_i \geq 0 \quad \text{for} \quad i=1,\ldots N \end{cases}$$

Primal optimization problem (L2-SVC)

$$\min_{\mathbf{w},b,\xi} \frac{\|\mathbf{w}\|^2}{2} + \frac{C}{2}\sum_{i=1}^{N}\xi_i^2 \quad \text{subject to} \quad y_i\left(\langle \mathbf{w},\mathbf{x}_i\rangle + b\right) \geq 1-\xi_i \quad \text{for } i=1,\ldots,N$$

Soft margin classification, dual optimization problem

Dual optimization problem (L1-SVC)

$$\max_{\boldsymbol{\alpha}} \; -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle + \sum_{i=1}^{N}\alpha_i$$

$$\text{subject to} \quad \begin{cases} \displaystyle\sum_{i=1}^{N}\alpha_i y_i = 0 \\ 0 \le \alpha_i \boxed{\le C} \quad \text{for} \quad i = 1,\ldots,N \end{cases}$$

Dual optimization problem (L2-SVC)

$$\max_{\boldsymbol{\alpha}} \; -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \left(\left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle \boxed{+\frac{1}{C}\delta_{ij}} \right) + \sum_{i=1}^{N}\alpha_i$$

$$\text{subject to} \quad \begin{cases} \displaystyle\sum_{i=1}^{N}\alpha_i y_i = 0 \\ \alpha_i \ge 0 \quad \text{for} \quad i = 1,\ldots,N \end{cases}$$

Soft margin classification, some remarks

Remark 1

The optimization problems are still **convex and quadratic programs**

Remark 2

C is a trade-off parameter

- ✓ Small value for C ▶ Constraint easily ignored ▶ Large margin
- ✓ Large value for C ▶ Constraint hard to ignore ▶ Narrow margin
- ✓ $C = \infty$ ▶ All constraints must be fulfilled (hard margin case)

Linear regression with Support Vector Machines

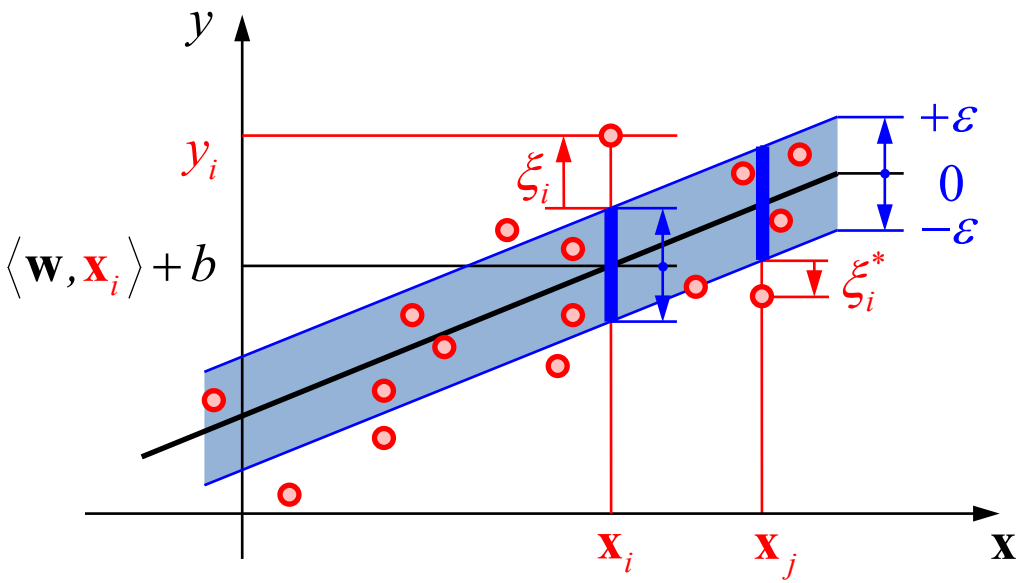
1 – SVR based on the ε-insensitive loss function

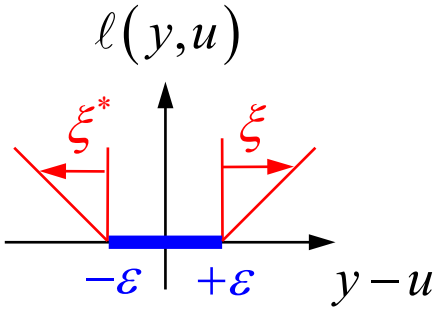
ε-insensitive SVM regression (ε-SVR) (Linear case)

Optimization problem (primal)

$$\min_{\mathbf{w},b} \frac{\|\mathbf{w}\|^2}{2} \quad \text{subject to} \quad \begin{cases} y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b \le \varepsilon & \text{for } i = 1,\ldots,N \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i \le \varepsilon & \text{for } i = 1,\ldots,N \end{cases}$$

Tacit assumption: all pairs $(\mathbf{x}_i, y_i)$ can be approximated with ε precision





No penalization of deviations lower than ε, linear penalization otherwise

ε-SVR with slack variables, primal optimization problem

Primal optimization problem (L1-ε-SVR)

$$\min_{\mathbf{w},b,\xi,\xi^*} \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right)$$

$$\text{subject to } \begin{cases} y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b \le \varepsilon + \xi_i & \text{for } i = 1,\ldots,N \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i \le \varepsilon + \xi_i^* & \text{for } i = 1,\ldots,N \\ \xi_i, \xi_i^* \ge 0 \quad \text{for } i = 1,\ldots N \end{cases}$$

Primal optimization problem (L2-ε-SVR)

$$\min_{\mathbf{w},b,\xi,\xi^*} \frac{\|\mathbf{w}\|^2}{2} + \frac{C}{2}\sum_{i=1}^{N}\left(\xi_i^2 + \xi_i^{*2}\right)$$

$$\text{subject to } \begin{cases} y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b \le \varepsilon + \xi_i & \text{for } i = 1,\ldots,N \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i \le \varepsilon + \xi_i^* & \text{for } i = 1,\ldots,N \end{cases}$$

Dual optimization problem for kernel ε-SVR

Dual optimization problem (L1-ε-SVR)

$$\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \; -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\alpha_i-\alpha_i^*\right)\left(\alpha_j-\alpha_j^*\right)\left\langle \mathbf{x}_i,\mathbf{x}_j\right\rangle+\sum_{i=1}^{N} y_i\left(\alpha_i-\alpha_i^*\right)-\varepsilon\sum_{i=1}^{N}\left(\alpha_i+\alpha_i^*\right)$$

$$\text{subject to} \quad \begin{cases} \sum_{i=1}^{N}\left(\alpha_i-\alpha_i^*\right)=0 \\ 0\le\alpha_i \le C \quad \text{for} \quad i=1,\ldots,N \\ 0\le\alpha_i^* \le C \quad \text{for} \quad i=1,\ldots,N \end{cases}$$

Dual optimization problem (L2-ε-SVR)

$$\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \; -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\alpha_i-\alpha_i^*\right)\left(\alpha_j-\alpha_j^*\right)\left(\left\langle \mathbf{x}_i,\mathbf{x}_j\right\rangle+\frac{1}{C}\delta_{ij}\right)+\sum_{i=1}^{N} y_i\left(\alpha_i-\alpha_i^*\right)-\varepsilon\sum_{i=1}^{N}\left(\alpha_i+\alpha_i^*\right)$$

$$\text{subject to} \quad \begin{cases} \sum_{i=1}^{N}\left(\alpha_i-\alpha_i^*\right)=0 \\ \alpha_i\ge 0 \quad \text{for} \quad i=1,\ldots,N \\ \alpha_i^*\ge 0 \quad \text{for} \quad i=1,\ldots,N \end{cases}$$

Decision function for kernel ε-SVR

Decision function (L1 & L2-ε-SVR)

$$\widetilde{f}(\mathbf{x}) = \left(\sum_{i \in \mathcal{I}_{\mathrm{sv}} \cup \mathcal{I}^*_{\mathrm{sv}}} \left(\alpha_i - \alpha_i^* \right) \langle \mathbf{x}_i, \mathbf{x} \rangle \right) + b$$

where $\mathcal{I}_{\mathrm{sv}} = \left\{ i : \alpha_i > 0 \right\}$ and $\mathcal{I}^*_{\mathrm{sv}} = \left\{ i : \alpha_i^* > 0 \right\}$

($\mathcal{I}_{\mathrm{sv}} \bigcup \mathcal{I}^*_{\mathrm{sv}}$ represents the set of indices of **support vectors**)

and b is conveniently obtained as a byproduct of any interior point optimization method.

Linear regression with Support Vector Machines

2 – Least squares SVM in regression

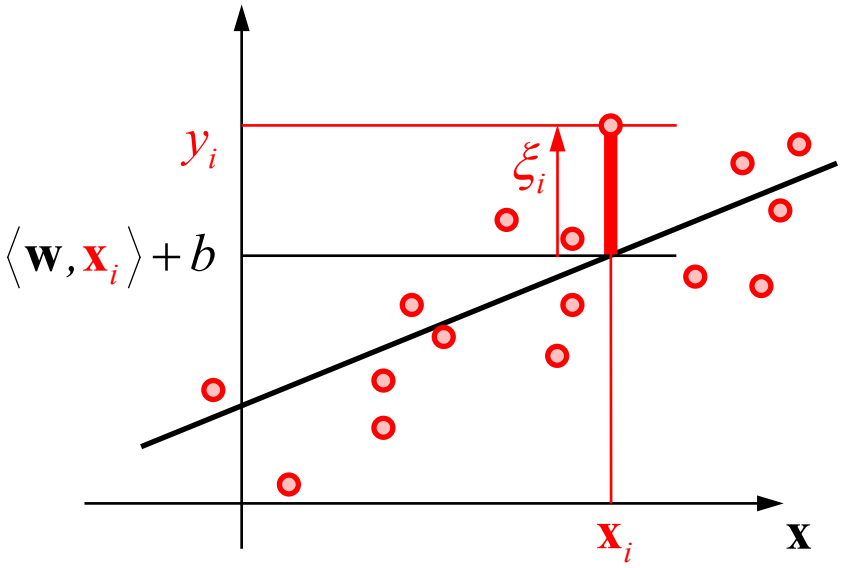
Least-squares SVM regression (LS-SVR) (Linear case)

Optimization problem (primal)

$$\min_{\mathbf{w},b,\xi} \frac{\|\mathbf{w}\|^2}{2} + \frac{C}{2}\sum_{i=1}^{N}\xi_i^2 \quad \text{subject to} \quad y_i = \langle \mathbf{w}, \mathbf{x}_i \rangle + b + \xi_i \quad \text{for } i = 1,\ldots,N$$

Residual sum of squares

Equality constraints



This problem is know as **ridge regression** (see e.g. Saunders et al. 1998)

Important remarks:

$C = \infty \quad \Leftrightarrow$ Ordinary least squares regression

$\langle \bullet, \bullet \rangle$ replaced by $k(\bullet, \bullet)$ $\Leftrightarrow$ Least squares SVM

Least-squares SVM regression (LS-SVR) (Linear case)

Lagrangian

$$L(\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\alpha}) = \frac{\|\mathbf{w}\|^2}{2} + \frac{C}{2}\sum_{i=1}^{N}\xi_i^2 - \sum_{i=1}^{N}\alpha_i\left(\langle \mathbf{w},\mathbf{x}_i\rangle + b + \xi_i - y_i\right)$$

$L(\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\alpha})$ minimized w.r.t. primal variables $\mathbf{w}$, b and $\boldsymbol{\xi}$,
and maximized w.r.t. dual variables α_i (saddle point solution)

Karush-Kuhn-Tucker conditions

$$\frac{\partial L(\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\alpha})}{\partial \mathbf{w}} = 0 \quad \blacktriangleright \quad \mathbf{w} = \sum_{i=1}^{N}\alpha_i\mathbf{x}_i \qquad (1)$$

$$\frac{\partial L(\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\alpha})}{\partial b} = 0 \quad \blacktriangleright \quad \sum_{i=1}^{N}\alpha_i = 0 \qquad (2)$$

$$\frac{\partial L(\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\alpha})}{\partial \boldsymbol{\xi}} = 0 \quad \blacktriangleright \quad \alpha_i = C\xi_i \quad \text{for} \quad i = 1,\ldots,N \qquad (3)$$

$$\frac{\partial L(\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = 0 \quad \blacktriangleright \quad \langle \mathbf{w},\mathbf{x}_i\rangle + b + \xi_i - y_i = 0 \quad \text{for} \quad i = 1,\ldots,N \qquad (4)$$

Least-squares SVM regression (LS-SVR)

Optimization problem (reformulated)

$$\left[\begin{array}{c|c} \mathbf{K}+C^{-1}\mathbf{I}_{N\times N} & \mathbf{1}_{N\times 1} \\ \hline \mathbf{1}_{N\times 1}^{\mathrm{T}} & 0 \end{array}\right]\left(\begin{array}{c} \boldsymbol{\alpha} \\ \hline b \end{array}\right)=\left(\begin{array}{c} \mathbf{y} \\ \hline 0 \end{array}\right)$$

by plugging (1), (2) and (3) into (4)

where $\mathbf{y}=(y_1,\ldots y_N)^{\mathrm{T}}$,

and $\mathbf{K}=\left[k_{ij}\right]_{1\le i,j\le N}$, in which $k_{ij}=\left\langle \mathbf{x}_i,\mathbf{x}_j\right\rangle$

The $(N+1)\times(N+1)$ system to solve is **linear** (so easier to solve than a QP)

A **global and unique** solution which is however **not sparse** in contrast to classical SVMs (all points are SVs here)

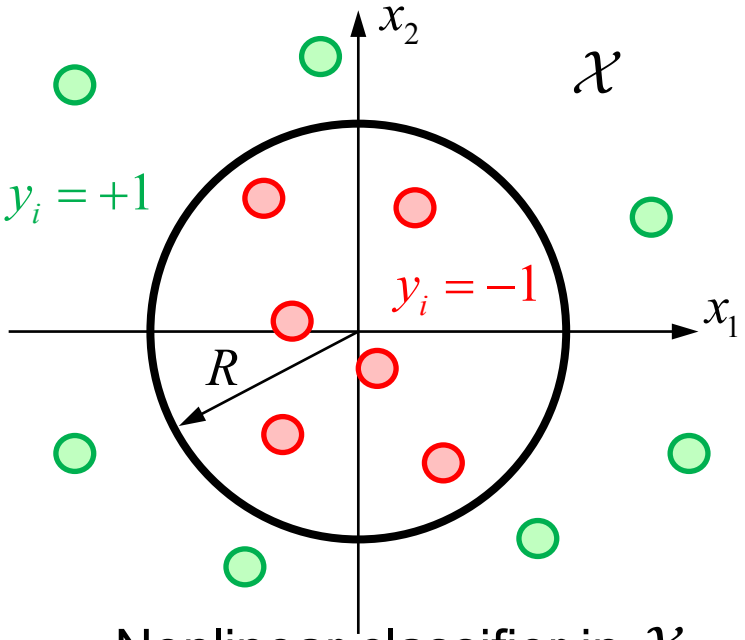
Decision function

$$\widetilde{f}(\mathbf{x})=\left(\sum_{i=1}^{N}\alpha_i\left\langle \mathbf{x}_i,\mathbf{x}\right\rangle\right)+b$$

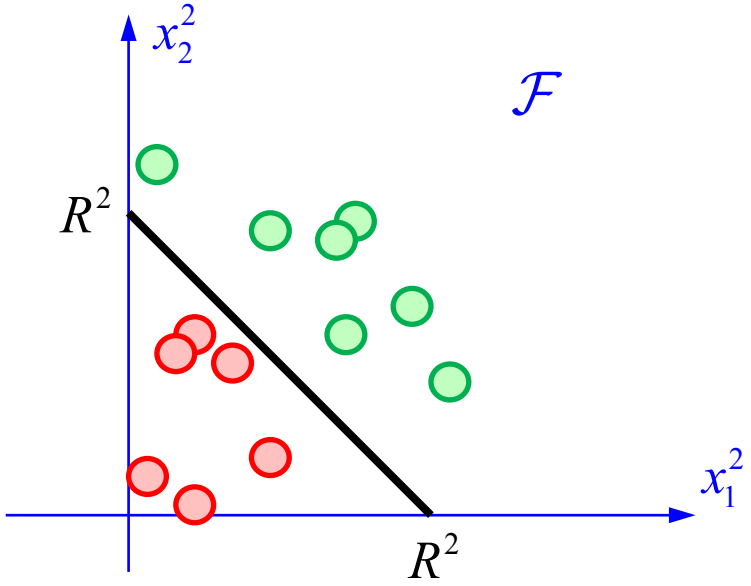
From linear to nonlinear SVM

1 – The kernel trick

From linear to nonlinear classifiers



Nonlinear classifier in $\mathcal{X}$



Linear classifier in $\mathcal{F}$

Mapping Φ :

$$\mathcal{X} \to \mathcal{F}$$

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

$\mathcal{F}$: **feature space**

$$f(x_1, x_2) = x_1^2 + x_2^2 - R^2$$

$$= \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} x_1^2 \\ x_2^2 \end{pmatrix} - R^2$$

$$= \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + b$$

where: $\mathbf{w} = \begin{pmatrix} 1 & 1 \end{pmatrix}^{\mathrm{T}}, \quad b = -R^2$

Reformulation of the primal problem (for L1-SVC)

- ✓ Given N data pairs: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \in \mathcal{X} \times \{-1, +1\}$
- ✓ Given a (nonlinear) feature map $\Phi : \mathcal{X} \to \mathcal{F}$ (e.g. $\mathcal{X} = \mathbb{R}^n$, $\mathcal{F} = \mathbb{R}^m$)
- ✓ Solve:

$$\min_{\mathbf{w},b,\xi} \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{N} \xi_i \quad \text{subject to} \quad \begin{cases} y_i\left(\langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle + b\right) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \quad \text{for } i = 1, \ldots, N$$

- ▶ The weight vector $\mathbf{w}$ now relates to the feature space $\mathcal{F}$
- ▶ Distances and angles are measured by the inner product $\langle \bullet, \bullet \rangle$ in $\mathcal{F}$
- ▶ The function $\tilde{f}$ is linear in $\mathcal{F}$: $\tilde{f}(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x})\rangle + b$

Do we need to explicitly define the mapping Φ?

The kernel "trick" (for L1-SVC)

Dual optimization problem

$$\max_{\boldsymbol{\alpha}} \; -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle + \sum_{i=1}^{N}\alpha_i \quad \text{s.t.} \begin{cases} \sum_{i=1}^{N}\alpha_i y_i = 0 \\ 0 \le \alpha_i \le C \;\; \text{for } i = 1,\ldots,N \end{cases}$$

▶ $$\max_{\boldsymbol{\alpha}} \; -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \underline{\left\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \right\rangle} + \sum_{i=1}^{N}\alpha_i \quad \text{s.t.} \begin{cases} \sum_{i=1}^{N}\alpha_i y_i = 0 \\ 0 \le \alpha_i \le C \;\; \text{for } i = 1,\ldots,N \end{cases}$$

Decision function

$$\widetilde{f}(\mathbf{x}) = \left(\sum_{i \in \mathcal{I}_{\text{sv}}} \alpha_i y_i \left\langle \mathbf{x}_i, \mathbf{x} \right\rangle \right) + b \qquad \blacktriangleright \qquad \widetilde{f}(\mathbf{x}) = \left(\sum_{i \in \mathcal{I}_{\text{sv}}} \alpha_i y_i \underline{\left\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \right\rangle} \right) + b$$

No, we only need $\left\langle \Phi(\bullet), \Phi(\bullet) \right\rangle$, the inner product in the feature space $\mathcal{F}$

$k(\mathbf{x}, \mathbf{x}') = \left\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \right\rangle$ is called the **kernel** between $\mathbf{x}$ and $\mathbf{x}'$

Dual and decision function for kernel SVC

Dual optimization problem (L1-SVC)

$$\max_{\boldsymbol{\alpha}} -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j)+\sum_{i=1}^{N}\alpha_i \quad \text{s.t.} \quad \begin{cases} \sum_{i=1}^{N}\alpha_i y_i = 0 \\ 0 \le \alpha_i \le C \ \text{ for } \ i=1,\ldots,N \end{cases}$$

Dual optimization problem (L2-SVC)

$$\max_{\boldsymbol{\alpha}} -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \left(k(\mathbf{x}_i,\mathbf{x}_j)+\frac{1}{C}\delta_{ij} \right)+\sum_{i=1}^{N}\alpha_i \quad \text{s.t.} \quad \begin{cases} \sum_{i=1}^{N}\alpha_i y_i = 0 \\ \alpha_i \ge 0 \ \text{ for } \ i=1,\ldots,N \end{cases}$$

Decision function (L1-SVC & L2-SVC)

$$\widetilde{f}(\mathbf{x}) = \left(\sum_{i\in\mathcal{S}} \alpha_i y_i k(\mathbf{x}_i,\mathbf{x}) \right) + b$$

From linear to nonlinear SVM

2 – The regularization viewpoint

SVM from the regularization viewpoint (1/2)

$$\min_{h \in \mathcal{H}_k,\, b \in \mathbb{R}} \; C \sum_{i=1}^{N} \ell\left(y_i, h(\mathbf{x}_i) + b\right) + \frac{1}{2}\|h\|_{\mathcal{H}_k}^2$$

where $\ell : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}^+$ is a **loss function** (convex w.r.t. second parameter but not necessarily differentiable),

and $C > 0$ is a **regularization constant**.

- The **first term** enforces the fit of the SVM model $\widetilde{f}(\mathbf{x}) = h(\mathbf{x}) + b$ to the training data
- The **second term** enforces a small norm of h in the RKHS which results in a sufficiently smooth solution

SVM from the regularization viewpoint (2/2)

$$\min_{h\in\mathcal{H}_k,\, b\in\mathbb{R}} \; C\sum_{i=1}^{N}\ell\left(y_i, h(\mathbf{x}_i)+b\right)+\frac{1}{2}\|h\|_{\mathcal{H}_k}^2$$

Solution h in the form $h(\mathbf{x})=\sum_{i=1}^{N} c_i k(\mathbf{x}_i,\mathbf{x})$

where $\mathbf{c}=(c_1,\ldots,c_N)$ is a vector of unknown expansion coefficients

and k is a **positive definite** kernel (reproducing kernel of the RKHS)

- $\|h\|_{\mathcal{H}_k}^2=\langle h,h\rangle_{\mathcal{H}_k}=\mathbf{c}^{\mathrm{T}}\mathbf{K}\mathbf{c}$

  where $\mathbf{K}=\left[k_{ij}\right]_{1\le i,j\le N}$ is the **Gram (or kernel) matrix**

  such that $k_{ij}=k(\mathbf{x}_i,\mathbf{x}_j)$

- $h(\mathbf{x}_i)=\sum_{j=1}^{N} c_j k(\mathbf{x}_j,\mathbf{x}_i)=(\mathbf{K}\mathbf{c})_i$

Unified formulation by Fenchel conjugates

Primal

$$\min_{c\in\mathbb{R}^N,\, b\in\mathbb{R}} \; C\sum_{i=1}^{N} \ell\left(y_i, (\mathbf{Kc})_i + b\right) + \tilde{g}\left(\sqrt{\mathbf{c}^{\mathrm{T}}\mathbf{Kc}}\right)$$

Convex optimization problem

where $\tilde{g}: \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$, $t \mapsto \frac{1}{2}t^2$ if $t \geq 0$, $t \mapsto +\infty$ otherwise

Dual (Fenchel duality)

$$\max_{\mathbf{p}\in\mathbb{R}^N,\, \mathbf{1}_N^{\mathrm{T}}\mathbf{p}=0} \; -C\sum_{i=1}^{N} \ell(y_i, \bullet)^*\left(-\frac{p_i}{C}\right) - \frac{1}{2}\mathbf{p}^{\mathrm{T}}\mathbf{Kp}$$

where $\mathbf{p} = (p_1, \ldots, p_N)$,

and $\ell(y_i, \bullet)^* : \mathbb{R} \to \mathbb{R}$ is the **Fenchel conjugate** of the convex function $\ell(y_i, \bullet) : u \mapsto \ell(y_i, u)$:

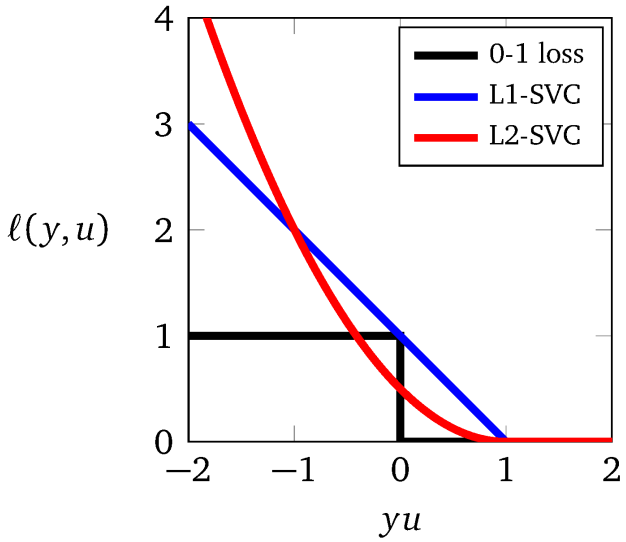
$$\ell(y_i, \bullet)^*(v) = \max_{u\in\mathbb{R}} \left\{ uv - \ell(y_i, u) \right\}$$

Loss functions and Fenchel conjugates (classification)

	Loss $\ell(y,u)$	Fenchel conjugate $\ell(y_i,\bullet)^*(v)$
L1-SVC	$(1-yu)_+$	vy_i if $vy_i \in [-1,0]$, $+\infty$ otherwise
L2-SVC	$\frac{1}{2}(1-yu)_+^2$	$\frac{1}{2}v^2 + vy_i$ if $vy_i \leq 0$, $+\infty$ otherwise

where $(x)_+ = \max(x,0)$

Def.: $(1-yu)_+$: Hinge loss function

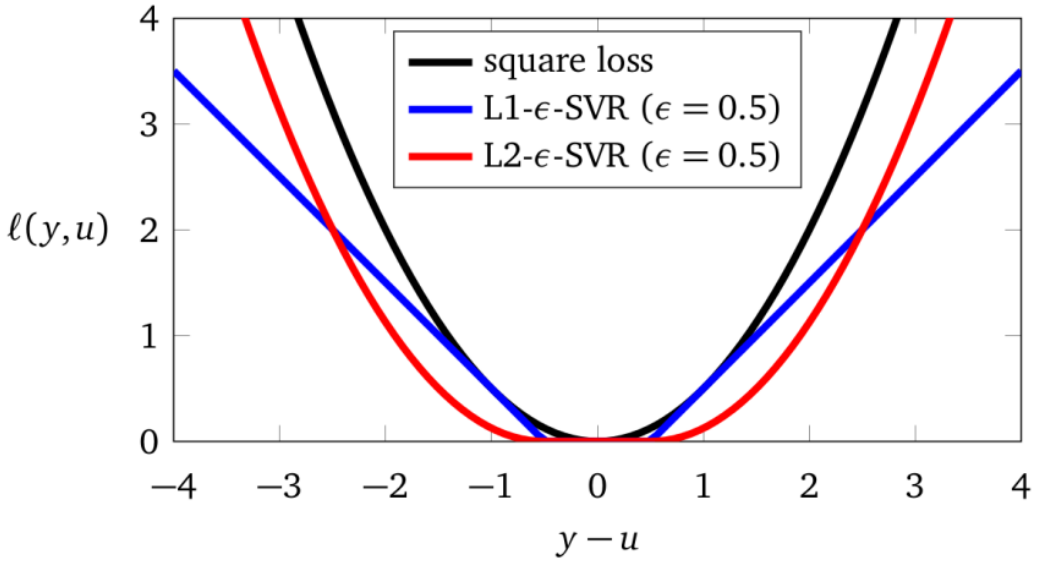


Loss functions and Fenchel conjugates (regression)

	Loss $\ell(y,u)$	Fenchel conjugate $\ell(y_i,\bullet)^*(v)$
LS-SVR, RN	$\frac{1}{2}(y-u)^2$	$\frac{1}{2}v^2+vy_i$
L1-ε-SVR	$\left(\lvert y-u\rvert-\varepsilon\right)_+$	$vy_i+\lvert v\rvert\varepsilon$ if $\lvert v\rvert\le 1$, $+\infty$ otherwise
L2-ε- SVR	$\frac{1}{2}\left(\lvert y-u\rvert-\varepsilon\right)_+^2$	$\frac{1}{2}v^2+vy_i+\lvert v\rvert\varepsilon$

where $(x)_+ = \max(x,0)$

Def.: $\left(\lvert y-u\rvert-\varepsilon\right)_+$: ε-insensitive loss function



Most usual kernels used in SVMs (1/2)

Gaussian RBF kernel

$$k(\mathbf{x},\mathbf{x}') = \exp(-\gamma \|\mathbf{x}-\mathbf{x}'\|^2)$$ where $\gamma \in \mathbb{R}_{>0}$

Most usual form: $k(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$ σ : **bandwidth parameter**

Important properties:

- ✓ stationary: $k(\mathbf{x},\mathbf{x}') = k_s(\mathbf{x}-\mathbf{x}')$ where $k_s(\mathbf{u}) = \exp(-\gamma\|\mathbf{u}\|^2)$
- ✓ isotropic (or radial): $k(\mathbf{x},\mathbf{x}') = k_r(r)$ where $k_r(r) = \exp(-\gamma r^2)$ and $r = \|\mathbf{x}-\mathbf{x}'\|$

Remarks:

- ✓ Good pick for smooth functions
- ✓ One single parameter to tune
- ✓ Most widely used kernel for SVMs

Most usual kernels used in SVMs (2/2)

Some other kernels

- ✓ **linear kernel**: $k(\mathbf{x},\mathbf{x}') = \langle \mathbf{x},\mathbf{x}' \rangle$

  … SVMs have been introduced with the linear kernel!

- ✓ **polynomial kernel**: $k(\mathbf{x},\mathbf{x}') = (c + \langle \mathbf{x},\mathbf{x}' \rangle)^d$ where $d \in \mathbb{N}_{>0}$ and $c \in \mathbb{R}_{>0}$

- ✓ **ν-Matérn kernel**: $k(\mathbf{x},\mathbf{x}') = \frac{1}{2^{\nu-1}\Gamma(\nu)} \left(\gamma\sqrt{2\nu}\,\|\mathbf{x}-\mathbf{x}'\|\right)^{\nu} K_{\nu}\left(\gamma\sqrt{2\nu}\,\|\mathbf{x}-\mathbf{x}'\|\right)$

  where $\nu \in \mathbb{R}_{\geq 1/2}$ is a regularity parameter (~ degree of smoothness) and $\theta \in \mathbb{R}_{>0}$.

  K_{ν} : modified Bessel function of second kind of order ν.

  This kernel has a closed form expression for $\nu = m + 1/2,\ m \in \mathbb{N}$.

- ✓ sigmoid, multiquadratic, inverse multiquadratic, thin plate splines, wavelets, …

Hyperparameter selection

Hyperparameters

Hyperparameters

Hyperparameters = Kernel parameter(s) + Other model parameters

$\boldsymbol{\theta}_k$ $\boldsymbol{\theta}_m$

Kernel parameters

- ✓ Gaussian RBF: γ
- ✓ Polynomial: c and d
- ✓ …

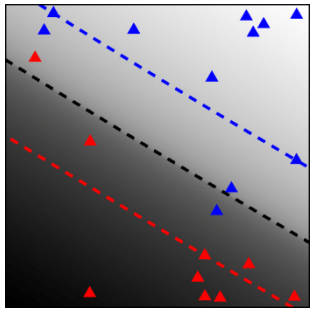
Other model parameters

- ✓ Regularization parameter: C
- ✓ Width of ε tube: ε (ε-SVR)

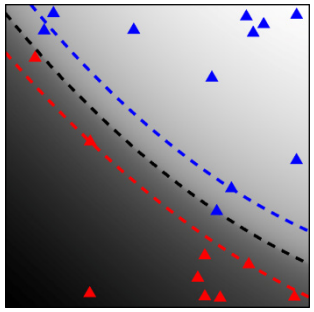
Gaussian RBF kernel – Role of hyperparameters

Role of γ ($C = 200$)

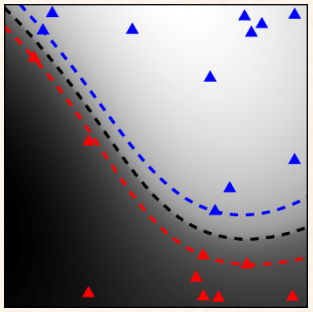
$\gamma = 0.002$



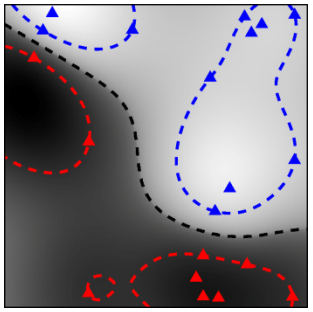
$\gamma = 0.02$



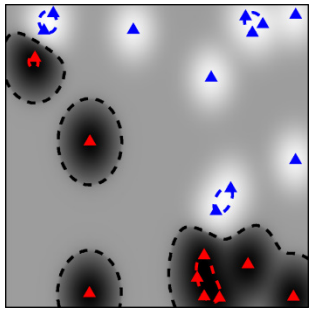
$\gamma = 0.2$



$\gamma = 2$

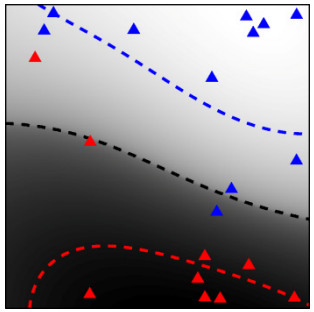


$\gamma = 20$

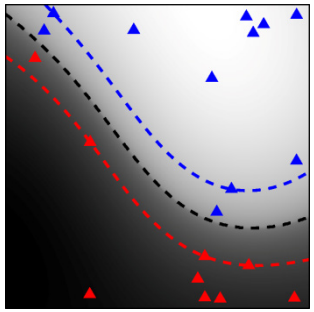


Role of C ($\gamma = 0.2$)

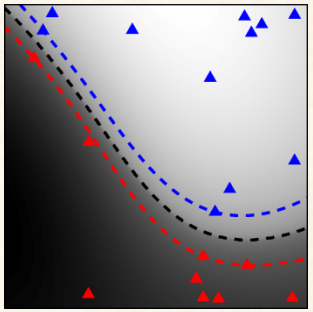
$C = 2$



$C = 20$



$C = 200$



Optimal selection of hyperparameters

Objective

$$(\boldsymbol{\theta}_m^*, \boldsymbol{\theta}_k^*) = \arg\min_{\boldsymbol{\theta}_m, \boldsymbol{\theta}_k} \widehat{\mathrm{Err}}_{\mathrm{gen}}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_k)$$

where $\widehat{\mathrm{Err}}_{\mathrm{gen}}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_k)$ is an (as accurate as possible) approximation of the generalization error for given values of parameters $\boldsymbol{\theta}_m$ and $\boldsymbol{\theta}_k$.

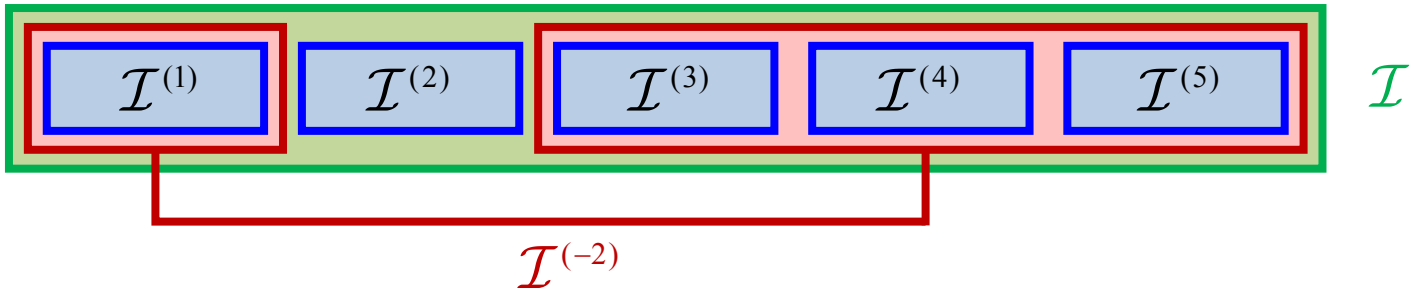
Note: the exact error (called true risk in SVMs) is not accessible!

Two key issues

- ✓ What can we take for $\widehat{\mathrm{Err}}_{\mathrm{gen}}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_k)$?
- ✓ How to efficiently solve the optimization problem?

Some notations for introducing CV and LOO-CV

- ✓ Indices of the whole training set: $\mathcal{I} = \{1, \ldots, N\}$
- ✓ K subsets $\mathcal{I}^{(k)}$ of $\mathcal{I}$ such that:
$$\begin{cases} K \in \{2, \ldots, N\} \\ \bigcup_{k=1}^{K} \mathcal{I}^{(k)} = \mathcal{I} \text{ and } \mathcal{I}^{(i)} \cap \mathcal{I}^{(j)} = \varnothing, \ \forall i \neq j \\ \#\mathcal{I}^{(1)} \approx \ldots \approx \#\mathcal{I}^{(K)} \approx \lceil N/k \rfloor \end{cases}$$
- ✓ Complementary set of $\mathcal{I}^{(k)}$ in $\mathcal{I}$: $\mathcal{I}^{(-k)} = \mathcal{I} \setminus \mathcal{I}^{(k)}$
- ✓ SVM trained on $\left\{ (\mathbf{x}_i, y_i), i \in \mathcal{I}^{(-k)} \right\}$: $\widetilde{f}^{(-k)}(\mathbf{x}) = \sum_{i \in \mathcal{I}^{(-k)}} c_i^{(-k)} k(\mathbf{x}_i, \mathbf{x}) + b^{(-k)}$
- ✓ Case $K = N$: $\mathcal{I}^{(-k)} = \{k\}$, $\mathcal{I}^{(-k)} = \{1, \ldots, N\} \setminus \{k\}$



K-fold cross validation (K-fold CV)

K-fold cross validation (K-fold CV)

$$\widehat{\mathrm{Err}}_{K\text{-CV},\,\ell}(\boldsymbol{\theta}) = \frac{1}{K}\sum_{k=1}^{K}\left[\frac{1}{\#\mathcal{I}^{(k)}}\sum_{i\in\mathcal{I}^{(k)}}\ell\left(y_i, \widetilde{f}^{(-k)}(\mathbf{x}_i;\boldsymbol{\theta})\right)\right]$$

where $\ell(y,u)$ is a selected loss function.

In practice, we often take $K=5$ or $K=10$.

The smaller K, the faster CV goes but:

- ✓ the training set is quite reduced compared to the whole one,
- ✓ the CV error has a high variance.

The above mentioned CV procedure may be repeated M times over other **random subsets** $\mathcal{I}^{(m,k)}$, $m=1,\ldots,M$, $k=1,\ldots,K$, for a reduced variance of $\widehat{\mathrm{Err}}_{K\text{-CV},\,\ell}$.

$$\widehat{\mathrm{Err}}_{K\text{-CV},\,\ell,\,M} = \frac{1}{M}\sum_{m=1}^{M}\left\{\frac{1}{K}\sum_{k=1}^{K}\left[\frac{1}{\#\mathcal{I}^{(m,k)}}\sum_{i\in\mathcal{I}^{(m,k)}}\ell\left(y_i, \widetilde{f}^{(-k)}(\mathbf{x}_i;\boldsymbol{\theta})\right)\right]\right\}$$

Leave-one-out cross-validation (LOO-CV)

Leave-one-out cross-validation (LOO-CV or simply LOO)

$$\mathrm{Err}_{\mathrm{LOO},\ell} = \widehat{\mathrm{Err}}_{N\text{-CV},\ell} = \frac{1}{N}\sum_{i=1}^{N} \ell\left(y_i, \widetilde{f}^{(-i)}(\mathbf{x}_i)\right)$$

LOO ≡ K-fold CV with $K = N$

where $\ell(y,u)$ is a selected loss function,

and $\widetilde{f}^{(-i)}(\mathbf{x}_i)$ is the predicted value at point $\mathbf{x}_i$ from the SVM model trained on $\{(\mathbf{x}_j, y_j)\ , j \in \{1,\ldots,N\} \setminus \{i\}\}$.

LOO-CV error is an **almost** unbiased estimate of the expected generalization error (Luntz & Brailovsky, 1969).

LOO-CV error is costly to evaluate (training of N SVMs from datasets composed of $(N-1)$ samples

▶ Bounds or approximations of the LOO error are often preferred.

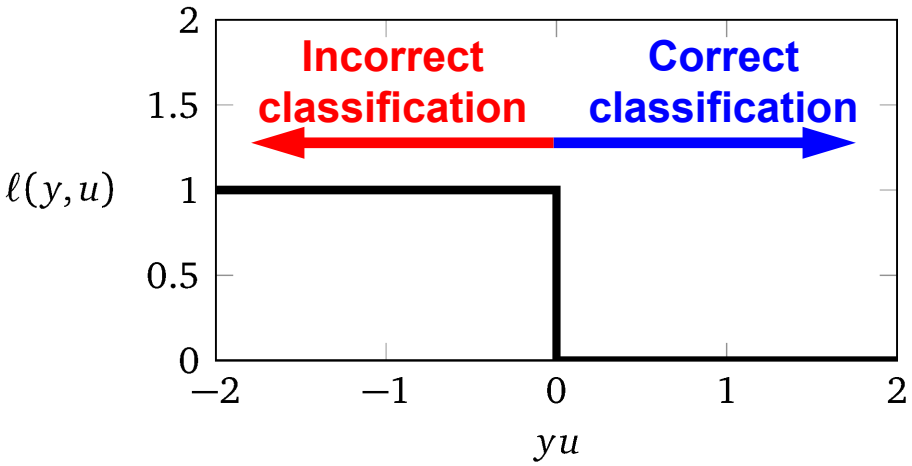
LOO error for SVM classification (SVC)

LOO error used for SVC (Vapnik & Chapelle 2000)

$$\mathrm{Err}_{\mathrm{LOO},\,\ell_{0-1}} = \frac{1}{N}\sum_{i=1}^{N} \ell_{0-1}\left(y_i, \widetilde{f}^{(-i)}(\mathbf{x}_i)\right) = \frac{\#\text{ misclassified points}}{N}$$

where: $\ell_{0-1}(y,u) = \Psi(-yu)$ (**misclassification** or **0-1 loss function**)

Ψ : Unit step function ($\Psi(x) = 1$ if $x > 0$, $\Psi(x) = 0$ otherwise)



LOO error for SVM regression (ε-SVR & LS-SVR)

- **LOO error used for ε-SVR (Chang & Lin 2005)**

$$\mathrm{Err}_{\mathrm{LOO},\,\ell_1} = \frac{1}{N}\sum_{i=1}^{N} \ell_1\left(y_i, \widetilde{f}^{(-i)}(\mathbf{x}_i)\right) = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \widetilde{f}^{(-i)}(\mathbf{x}_i)\right|$$

where $\ell_1(y,u) = |y-u|$ (ℓ_1 **loss function**)

- **LOO error used for LS-SVR (Cawley & Talbot 2004)**

$$\mathrm{Err}_{\mathrm{LOO},\,\ell_2} = \frac{1}{N}\sum_{i=1}^{N} \ell_2\left(y_i, \widetilde{f}^{(-i)}(\mathbf{x}_i)\right) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \widetilde{f}^{(-i)}(\mathbf{x}_i)\right)^2$$

where $\ell_2(y,u) = (y-u)^2$ (**square loss function**)

Mean Square Error (MSE)

A simple (but loose) bound for hard margin SVC

$$\mathrm{Err}_{\mathrm{LOO},\,\ell_{0\text{-}1}} = \frac{1}{N}\sum_{i=1}^{N}\Psi\left(-y_i\,\widetilde{f}^{(-i)}(\mathbf{x}_i)\right)$$

$$= \frac{1}{N}\sum_{i=1}^{N}\Psi\left(\,-y_i\,\widetilde{f}(\mathbf{x}_i)\;+\;y_i\left\{\widetilde{f}(\mathbf{x}_i)-\widetilde{f}^{(-i)}(\mathbf{x}_i)\right\}\;\right)$$

For non support vector points ($i \notin \mathcal{I}_{\mathrm{sv}}$):

$$\widetilde{f}^{(-i)}(\mathbf{x}) = \widetilde{f}(\mathbf{x}) = \left(\sum_{i\in\mathcal{I}_{\mathrm{sv}}}\alpha_i y_i k(\mathbf{x}_i,\mathbf{x})\right)+b \quad \blacktriangleright \quad \widetilde{f}(\mathbf{x}_i)-\widetilde{f}^{(-i)}(\mathbf{x}_i)=0$$

For hard margin SVC, we have: $y_i\,\widetilde{f}(\mathbf{x}_i)>1$ for $i=1,\ldots,N$

(all points are correctly classified and none of them lies in the margin)

▶ $-y_i\,\widetilde{f}(\mathbf{x}_i)<-1$ for $i\notin\mathcal{I}_{\mathrm{sv}}$ (in particular)

▶ $$\mathrm{Err}_{\mathrm{LOO},\,\ell_{0\text{-}1}} = \frac{1}{N}\sum_{i\in\mathcal{I}_{\mathrm{sv}}}\Psi\left(\,-y_i\,\widetilde{f}(\mathbf{x}_i)\;+\;y_i\left\{\widetilde{f}(\mathbf{x}_i)-\widetilde{f}^{(-i)}(\mathbf{x}_i)\right\}\;\right)$$

▶ $$\mathrm{Err}_{\mathrm{LOO},\,\ell_{0\text{-}1}} \le \frac{\#\mathcal{I}_{\mathrm{sv}}}{N}$$

How to practically define $\mathcal{I}_{\mathrm{sv}}$?

$\alpha_i = 0$ ▶ $|\alpha_i| < \delta$ where $\delta \ll 1$

Some other bounds for SVC LOO error

- **Jaakkola-Haussler bound** (hard margin SVC without bias term)

$$y_i \left\{ \widetilde{f}(\mathbf{x}_i) - \widetilde{f}^{(-i)}(\mathbf{x}_i) \right\} \leq \alpha_i k(\mathbf{x}_i, \mathbf{x}_i)$$

  - $$\mathrm{Err}_{\mathrm{LOO},\, \ell_{0\text{-}1}} \leq \frac{1}{N} \sum_{i=1}^{N} \Psi\left(-1 + \alpha_i k(\mathbf{x}_i, \mathbf{x}_i) \right)$$

- **Opper-Winter bound** (hard margin SVC without bias term)

We have: $$\widetilde{f}^{(-i)}(\mathbf{x}) = \left(\sum_{j \in \mathcal{I}^{(-i)}} \alpha_j^{(-i)} y_j k(\mathbf{x}_j, \mathbf{x}) \right) + b^{(-i)}$$

Under the assumption $\mathcal{I}_{\mathrm{sv}}{}^{(-i)} = \mathcal{I}_{\mathrm{sv}}$ where $\mathcal{I}_{\mathrm{sv}}{}^{(-i)} = \left\{ j \in \mathcal{I}^{(-i)} : \alpha_j^{(-i)} \neq 0 \right\}$,

we have: $$y_i \left\{ \widetilde{f}(\mathbf{x}_i) - \widetilde{f}^{(-i)}(\mathbf{x}_i) \right\} = \frac{\alpha_i}{\left(\mathbf{K}_{\mathrm{sv}}^{-1} \right)_{ii}}$$ where $\mathbf{K}_{\mathrm{sv}} = \left[k(\mathbf{x}_i, \mathbf{x}_j) \right]_{i,j \in \mathcal{I}_{\mathrm{sv}}}$

  - $$\mathrm{Err}_{\mathrm{LOO},\, \ell_{0\text{-}1}} \leq \frac{1}{N} \sum_{i=1}^{N} \Psi\left(-1 + \frac{\alpha_i}{\left(\mathbf{K}_{\mathrm{sv}}^{-1} \right)_{ii}} \right)$$

Span approximation for SVC LOO error (1/2)

Vapnik & Chapelle span approximation (L1 & L2-SVC)

The support vectors for L1-SVC are cast into two categories:

- ✓ **unbounded** SVs: $\mathcal{I}_{\text{usv}} = \{ i \ : \ 0 < \alpha_i < C \}$
- ✓ **bounded** SVs: $\mathcal{I}_{\text{bsv}} = \{ i \ : \ \alpha_i = C \}$

Under the assumption $\mathcal{I}_{\text{usv}}^{(-i)} = \mathcal{I}_{\text{usv}}$ **(and** $\mathcal{I}_{\text{bsv}}^{(-i)} = \mathcal{I}_{\text{bsv}}$ **for L1-SVC),**

where $\mathcal{I}_{\text{usv}}^{(-i)} = \left\{ \ j \in \mathcal{I}^{(-i)} : \ 0 < \alpha_j^{(-i)} < C \ \right\}$ and $\mathcal{I}_{\text{bsv}}^{(-i)} = \left\{ \ j \in \mathcal{I}^{(-i)} : \ \alpha_j^{(-i)} = C \ \right\}$ for L1-SVC,

and $\mathcal{I}_{\text{usv}}^{(-i)} = \left\{ \ j \in \mathcal{I}^{(-i)} : \ \alpha_j^{(-i)} > 0 \ \right\}$ for L2-SVC,

we have: $y_i \left\{ \widetilde{f}(\mathbf{x}_i) - \widetilde{f}^{(-i)}(\mathbf{x}_i) \right\} = \alpha_i S_i^2$

Span approximation for SVC LOO error (2/2)

Vapnik & Chapelle span approximation (L1 & L2-SVC)

Under the assumption $\mathcal{I}_{\text{usv}}^{(-i)} = \mathcal{I}_{\text{usv}}$ **(and** $\mathcal{I}_{\text{bsv}}^{(-i)} = \mathcal{I}_{\text{bsv}}$ **for L1-SVC),**

we have: $$y_i \left\{ \widetilde{f}(\mathbf{x}_i) - \widetilde{f}^{(-i)}(\mathbf{x}_i) \right\} = \alpha_i S_i^2$$

S_i represents the distance between $\Phi(\mathbf{x}_i)$ and the set Λ_i such that:

$$\Lambda_i = \left\{ \sum_{j \in \mathcal{I}_{\text{usv}},\, j \neq i} \lambda_j \Phi(\mathbf{x}_j) \;:\; \lambda_j \in \mathbb{R} \;,\; \sum_{j \in \mathcal{I}_{\text{usv}},\, j \neq i} \lambda_j = 1 \right\}$$

S_i is called the **span of the SV** $\mathbf{x}_i$.

We therefore have: $S_i = \text{dist}\left(\Phi(\mathbf{x}_i), \Lambda_i\right) = \min_{\mathbf{z} \in \Lambda_i} \left\| \Phi(\mathbf{x}_i) - \mathbf{z} \right\|$

$$\widetilde{\text{Err}}_{\text{LOO},\,\ell_{0\text{-}1}} = \frac{1}{N} \sum_{i=1}^{N} \Psi\left(-y_i \widetilde{f}(\mathbf{x}_i) + \alpha_i S_i^2 \right)$$

$$= \frac{1}{N} \#\left\{ i \;:\; \alpha_i S_i^2 \geq y_i \widetilde{f}(\mathbf{x}_i) \right\}$$

This expression is an approximation, the main assumption has no chance to be met!

Practical calculation of the span approximation for SVC

Calculation of Vapnik & Chapelle span approximation

- ✓ Unbounded SVs for L1-SVC ($i \in \mathcal{I}_{\text{usv}}$)
  or all SVs for L2-SVC ($i \in \mathcal{I}_{\text{sv}}$ $(\equiv \mathcal{I}_{\text{usv}})$)

$$S_i^2 = \frac{1}{\left(\widetilde{\mathbf{K}}_{\text{usv}}^{-1}\right)_{ii}} \quad \text{where} \quad \widetilde{\mathbf{K}}_{\text{usv}} = \begin{bmatrix} \mathbf{K}_{\text{usv}} & \mathbf{1} \\ \mathbf{1}^{\text{T}} & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{K}_{\text{usv}} = \left[k(\mathbf{x}_i, \mathbf{x}_j) \right]_{i,j \in \mathcal{I}_{\text{usv}}}$$

- ✓ Bounded SVs for L1-SVC ($i \in \mathcal{I}_{\text{bsv}}$)

$$S_i^2 = k(\mathbf{x}_i, \mathbf{x}_i) - \mathbf{v}_i^{\text{T}} \widetilde{\mathbf{K}}_{\text{usv}}^{-1} \mathbf{v}_i \quad \text{where} \quad \mathbf{v}_i = \left(\left(k(\mathbf{x}_i, \mathbf{x}_j), j \in \mathcal{I}_{\text{usv}} \right), 1 \right)^{\text{T}}$$

- ✓ Non SVs ($i \in \overline{\mathcal{I}}_{\text{sv}}$)

$$S_i^2 = 0$$

Span approximation for ε-SVR LOO error

Span of SV $\mathbf{x}_i$

$$S_i^2 = \frac{1}{\left(\widetilde{\mathbf{K}}_{\text{usv}}^{-1}\right)_{ii}} \quad \text{where} \quad \widetilde{\mathbf{K}}_{\text{usv}} = \begin{bmatrix} \mathbf{K}_{\text{usv}} & \mathbf{1} \\ \mathbf{1}^{\text{T}} & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{K}_{\text{usv}} = \left[k(\mathbf{x}_i, \mathbf{x}_j) \right]_{i,j \in \mathcal{I}_{\text{usv}}}$$

where: $\mathcal{I}_{\text{usv}} = \left\{ i \ : \ 0 < \alpha_i + \alpha_i^* < C \right\}$ for L1-ε-SVR

$\mathcal{I}_{\text{usv}} = \left\{ i \ : \ \alpha_i + \alpha_i^* > 0 \right\}$ for L2-ε-SVR

Span approximation (L1-ε-SVR)

$$\widetilde{\text{Err}}_{\text{LOO},\ell_1} = \varepsilon + \frac{1}{N} \sum_{i \in \mathcal{I}_{\text{usv}}} (\alpha_i + \alpha_i^*) S_i^2 + \frac{1}{N} \sum_{i \in \mathcal{I}_{\text{usv}}} (\xi_i + \xi_i^*)$$

Span approximation (L2-ε-SVR)

$$\widetilde{\text{Err}}_{\text{LOO},\ell_1} = \varepsilon + \frac{1}{N} \sum_{i \in \mathcal{I}_{\text{usv}}} (\alpha_i + \alpha_i^*) S_i^2$$

These expressions are under the assumption that

$$\mathcal{I}_{\text{usv}}^{(-i)} = \mathcal{I}_{\text{usv}}$$

They are therefore approximations, this assumption being rarely met

LOO error for SVM regression (LS-SVR)

Recall

$$\left[\begin{array}{c|c} \mathbf{K}+C^{-1}\mathbf{I}_{N\times N} & \mathbf{1}_{N\times 1} \\ \hline \mathbf{1}_{N\times 1}^{\mathrm{T}} & 0 \end{array}\right]\left(\begin{array}{c} \boldsymbol{\alpha} \\ \hline b \end{array}\right)=\left(\begin{array}{c} \mathbf{y} \\ \hline 0 \end{array}\right)$$

where $\mathbf{y}=\left(y_1,\ldots y_N\right)^{\mathrm{T}}$ and $\mathbf{K}=\left[k(\mathbf{x}_i,\mathbf{x}_j)\right]_{1\le i,j\le N}$

Problem setup (which we do not need to solve!)

For each $i=1,\ldots N$, solve the following $N\times N$ linear systems:

$$\left[\begin{array}{c|c} \mathbf{K}^{(-ii)}+C^{-1}\mathbf{I}_{(N-1)\times(N-1)} & \mathbf{1}_{(N-1)\times 1} \\ \hline \mathbf{1}_{(N-1)\times 1}^{\mathrm{T}} & 0 \end{array}\right]\left(\begin{array}{c} \boldsymbol{\alpha}^{(-i)} \\ \hline b^{(-i)} \end{array}\right)=\left(\begin{array}{c} \mathbf{y}^{(-i)} \\ \hline 0 \end{array}\right)$$

We are striking out the i^{th} line and i^{th} column of the full system of equations

where $\mathbf{v}^{(-i)}$: $\mathbf{v}$ vector with i^{th} element removed
and $\mathbf{M}^{(-ii)}$: $\mathbf{M}$ matrix with i^{th} row and i^{th} column removed

$$\widetilde{f}^{(-i)}(\mathbf{x})=\left(\sum_{j=1}^{N-1}\alpha_j^{(-i)}k(\mathbf{x}_j,\mathbf{x})\right)+b^{(-i)}$$

LOO error for SVM regression (LS-SVR)

Recall

$$\left[\begin{array}{c|c} \mathbf{K}+C^{-1}\mathbf{I}_{N\times N} & \mathbf{1}_{N\times 1} \\ \hline \mathbf{1}_{N\times 1}^{\mathrm{T}} & 0 \end{array}\right]\left(\begin{array}{c} \boldsymbol{\alpha} \\ \hline b \end{array}\right)=\left(\begin{array}{c} \mathbf{y} \\ \hline 0 \end{array}\right) \qquad \text{Notation:} \quad \mathbf{M}=\left[\begin{array}{cc} \mathbf{K}+C^{-1}\mathbf{I}_{N\times N} & \mathbf{1}_{N\times 1} \\ \mathbf{1}_{N\times 1}^{\mathrm{T}} & 0 \end{array}\right]$$

where $\mathbf{y}=\left(y_1,\ldots y_N\right)^{\mathrm{T}}$ and $\mathbf{K}=\left[k(\mathbf{x}_i,\mathbf{x}_j)\right]_{1\le i,j\le N}$

LOO error

$$y_i-\tilde{f}^{(-i)}(\mathbf{x}_i)=\frac{\alpha_i}{\left(\mathbf{M}^{-1}\right)_{ii}}$$

$$\mathrm{Err}_{\mathrm{LOO},\ell_2}=\frac{1}{N}\sum_{i=1}^{N}\ell_2\left(y_i,\tilde{f}^{(-i)}(\mathbf{x}_i)\right)=\frac{1}{N}\sum_{i=1}^{N}\left(y_i-\tilde{f}^{(-i)}(\mathbf{x}_i)\right)^2=\frac{1}{N}\sum_{i=1}^{N}\left(\frac{\alpha_i}{\left(\mathbf{M}^{-1}\right)_{ii}}\right)^2$$

PRESS (Allen, 1971)

where $\ell_2(y,u)=(y-u)^2$ (square loss function)

This is an exact expression!

Approximation of the generalization error

Objective

$$(\boldsymbol{\theta}_{\mathrm{m}}^{*}, \boldsymbol{\theta}_{\mathrm{k}}^{*}) = \arg\min_{\boldsymbol{\theta}_{\mathrm{m}}, \boldsymbol{\theta}_{\mathrm{k}}} \widehat{\mathrm{Err}}_{\mathrm{gen}}(\boldsymbol{\theta}_{\mathrm{m}}, \boldsymbol{\theta}_{\mathrm{k}})$$

Two key issues

- ✓ **What can we take for $\widehat{\mathrm{Err}}_{\mathrm{gen}}(\boldsymbol{\theta}_{\mathrm{m}}, \boldsymbol{\theta}_{\mathrm{k}})$?**
- ✓ How to efficiently solve the optimization problem?

Approximations used for $\widehat{\mathrm{Err}}_{\mathrm{gen}}(\boldsymbol{\theta}_{m}, \boldsymbol{\theta}_{k})$

- ✓ **Real K-fold CV** (most usual practice for SVMs)

$$\widehat{\mathrm{Err}}_{\mathrm{gen}}(\boldsymbol{\theta}_{m}, \boldsymbol{\theta}_{k}) = \widehat{\mathrm{Err}}_{K\text{-CV},\ell}$$

or $\widehat{\mathrm{Err}}_{K\text{-CV},\ell,M}$ (if we can afford it!)

- ✓ Real LOO CV (too costly, not used in real applications)
- ✓ **LOO error approximations** (or exact expression if available!)

$$\widehat{\mathrm{Err}}_{\mathrm{gen}}(\boldsymbol{\theta}_{m}, \boldsymbol{\theta}_{k}) = \widetilde{\mathrm{Err}}_{\mathrm{LOO},\ell}$$

Finding optimal values for hyperparameters

Objective

$$(\boldsymbol{\theta}_m^*, \boldsymbol{\theta}_k^*) = \arg\min_{\boldsymbol{\theta}_m, \boldsymbol{\theta}_k} \widehat{\mathrm{Err}}_{\mathrm{gen}}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_k)$$

Two key issues

- ✓ What can we take for $\widehat{\mathrm{Err}}_{\mathrm{gen}}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_k)$?
- ✓ **How to efficiently solve the optimization problem?**

Techniques used for solving the optimization problem

- ✓ **Deterministic search strategies** (grid search essentially)
- ✓ **Stochastic optimization methods**

Grid search technique

For each parameter θ_i of the SVM model ($\boldsymbol{\theta}_{\mathrm{m}}$ or $\boldsymbol{\theta}_{\mathrm{k}}$), $i = 1,\ldots,n_{\boldsymbol{\theta}}$, define a **range** $[a_{\theta_i}, b_{\theta_i}]$ for exploration.
In practice, a range in **log scale** is often preferred: $[a_{\log\theta_i}, b_{\log\theta_i}] \equiv [a_i, b_i]$

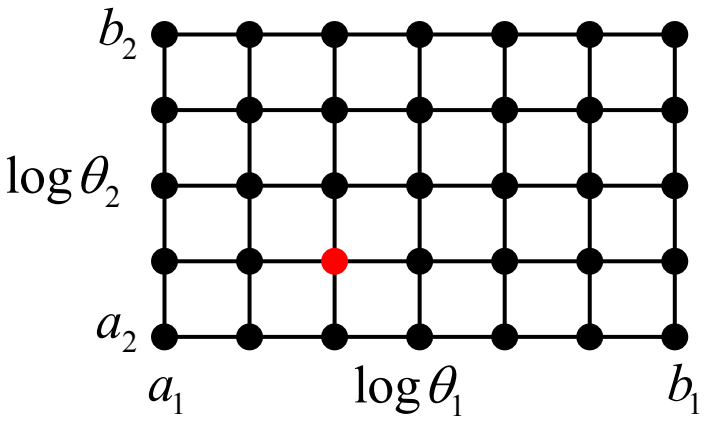
Select a **level of discretization** n_i for each range for **equally spaced values** ($\log\theta_{i,j}$, $j = 1,\ldots,n_i$) such that $\log\theta_{i,1} = a_i$ and $\log\theta_{i,n_i} = b_i$.

The optimal solution is given by:

$$\boldsymbol{\theta}^* = \arg\min_{\log\boldsymbol{\theta}_j \in \mathcal{G}} \widehat{\mathrm{Err}}_{\mathrm{gen}}(\boldsymbol{\theta}_j)$$

where $\mathcal{G}$ represents the **search grid**:

$$\mathcal{G} = \left\{ (\log\theta_{1,i_1},\ldots,\log\theta_{n_{\boldsymbol{\theta}},i_{n_{\boldsymbol{\theta}}}}) \ , \ i_1 \in \{1,\ldots,n_1\}, \ \ldots, \ i_{n_{\boldsymbol{\theta}}} \in \{1,\ldots,n_{n_{\boldsymbol{\theta}}}\} \right\}$$



Some comments about grid search technique

Example

For a problem with:

- ✓ 3 hyperparameters to tune (e.g. C, ε and γ for ε-SVR with a Gaussian RBF kernel),
- ✓ a $20 \times 20 \times 20$ grid for parameter combinations,
- ✓ the use of a 5-fold CV.

▶ $20 \times 20 \times 20 \times 5$ = Need the training of 40,000 SVRs!

Main drawbacks of grid search

- ✓ The parameter space is discretized
  ▶ Lack of accuracy of the solution
- ✓ The method does not scale well with the number of parameters

Alternatives to grid search technique

Gradient-based optimization methods

$\widehat{\mathrm{Err}}_{K\text{-CV},\ell}$ and $\mathrm{Err}_{\mathrm{LOO},\ell}$ are very often **noisy** (with **multi-extrema**).
The predicted errors are in addition **discontinuous** for SVC due to the loss function used for defining the error (0-1 loss for SVCs).
▶ Gradient-based searching techniques do not work!

Alternative: Gradient-based optimization method + smoothed version of the LOO error (Chapelle et al. 2002).

Stochastic optimization methods

- ✓ Simulated annealing (Pai & Hong 2006, Lin et al. 2008),
- ✓ Genetic algorithms (Pai 2006, Chen 2007),
- ✓ Particle swarm optimization (Lin et al. 2008, Fei et al. 2009),
- ✓ Particle filter (Wei et al. 2013),
- ✓ **Cross-entropy method** (Bourinet 2013, Bourinet 2016).

Solving the QP optimization problem

We usually solve the dual optimization problem (some alternatives exist).

SVM kernel matrix is dense (algorithms assuming sparsity are excluded).

Size of the training set ▶ Small-scale or large-scale learning

◢ Two main types of algorithms

- ✓ **Sequential minimal optimization (SMO) algorithm**

  Introduced by Platt (Platt 1999)
  Decomposition method working with minimal working set size: 2 α_i's
  Many variants including the one used in LIBSVM (Fan et al. 2005)
  Not very accurate for large C (Bottou & Lin 2007)

- ✓ **Interior point algorithms** (Nesterov & Nemirovskii 1993, Nocedal & Wright 2006)

  LOQO (Vanderbei 1999)
  For small to moderately large problems (~1,000 training samples)
  Reliable methods with high precision

Reliability assessment based on Support Vector Machines

Surrogate-based reliability assessment

Prior works

- ✓ Polynomial RS (1990-2000+), polynomial MLS (2008-2010)
- ✓ PCE (2002+)
- ✓ ANN (2000+)
- ✓ SVM (2002+)
- ✓ Kriging (2005 and 2008+)

SVM-based approaches

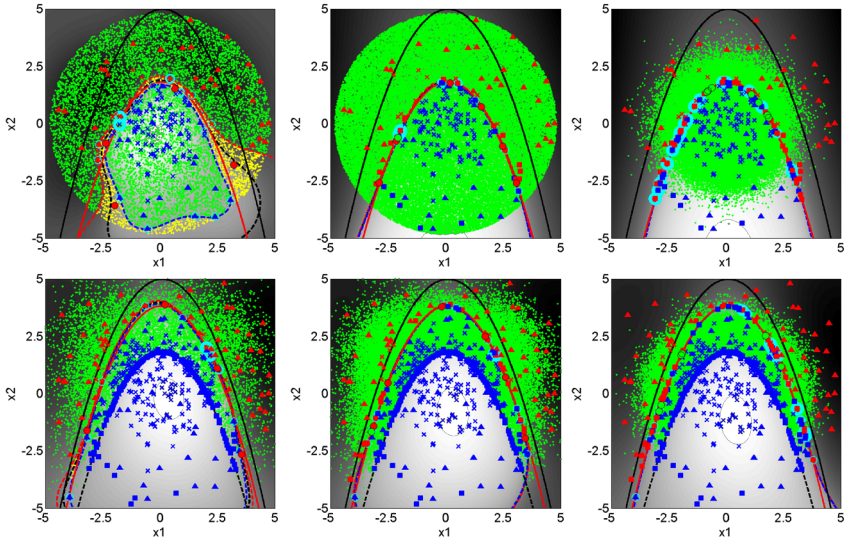
- ✓ Seminal works of Rocco & Moreno 2002, Hurtado 2004
- ✓ Mostly classification but also regression
- ✓ Almost no details about hyperparameter selection, QP solvers

Kriging-based approaches

- ✓ Seminal work of Keymaz 2005
- ✓ EGRA (Bichon et al. 2008), AK-MCS (Echard 2011), SUR (Bect et al. 2012)
- ✓ Mainly differ in terms of the criterion used for doe enrichment

Short presentation of ^{2}SMART (Bourinet et al. 2011)

- ✓ Sampling and training in the standard normal space
- ✓ Training of an L1-SVC (classification) applied to each intermediate LSF such as defined in SS
- ✓ SVM surrogate model trained with grid selection and 3-fold cross validation, LIBSVM (SMO) as solver



- ✓ Selection of new training points with a coarse to fine adaptive strategy (selection from points generated by MCMC (m-M): clustering of points in the margin, class-instable points, closest point to SVC classifier)

Adaptive Support Vector Regression (ASVR)

Main characteristics of the ASVR method (Bourinet 2016)

- ✓ Sampling and training **in the standard normal space**
- ✓ Failure probability estimate : $p_{\mathrm{f}}^{\mathrm{ASVR}} = \mathbb{E}_{\varphi_n}\left[1_{\widetilde{\mathcal{F}}_{\mathbf{u}}}(\mathbf{U})\right]$

  where $\widetilde{\mathcal{F}}_{\mathbf{u}} = \left\{\mathbf{u} \in \mathbb{R}^n : \widetilde{G}_{s_{\mathrm{final}}}(\mathbf{u}) \leq 0\right\}$
- ✓ Learning intermediate LSFs with high accuracy is a waste of time

  ► **Fast exploration with training samples which progressively reach and populate the failure domain**
- ✓ Only the sign of $G(\mathbf{u})$ is used in SVC, the exact value is not accounted for (loss of information)

  ► Use of L1-ε-SVR (**regression**) in ASVR method
- ✓ QP solved by an **interior point method**

  ► High accuracy on α_i and α_i^*
- ✓ True k-fold cross validation avoided

  ► Use of the **span approximation of the LOO error**
- ✓ Inaccuracy of grid search for hyperparameter selection

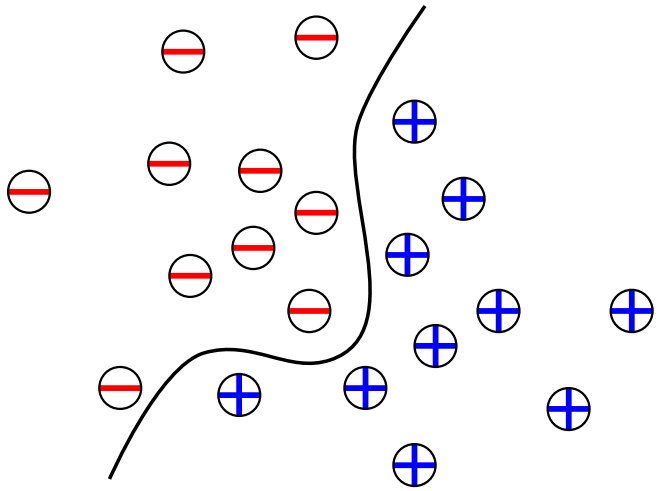
  ► **Stochastic search with the CE method**

ASVR method (1)

$\widetilde{G}_1(\mathbf{u}) = y_{\text{th};1}$

$\oplus$: $\mathbf{u}_i : G(\mathbf{u}_i) > y_{\text{th};1}$

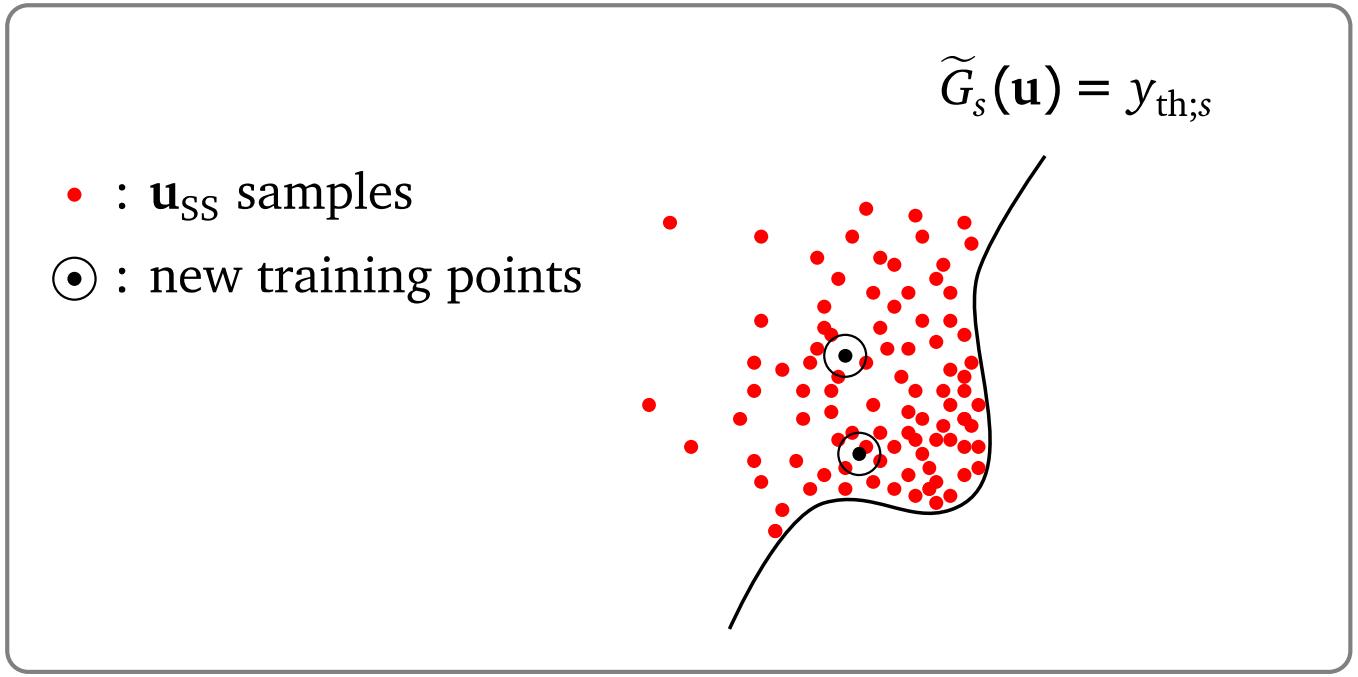
$\ominus$: $\mathbf{u}_i : G(\mathbf{u}_i) < y_{\text{th};1}$



Initial training set, SVR surrogate $\widetilde{G}_1$

Selection of 1st intermediate level $y_{\text{th},\,s} = \text{median}\left(\mathbf{g}_{\mathcal{I}_{\text{train}}\,;\,1,\ldots,N}\right)$

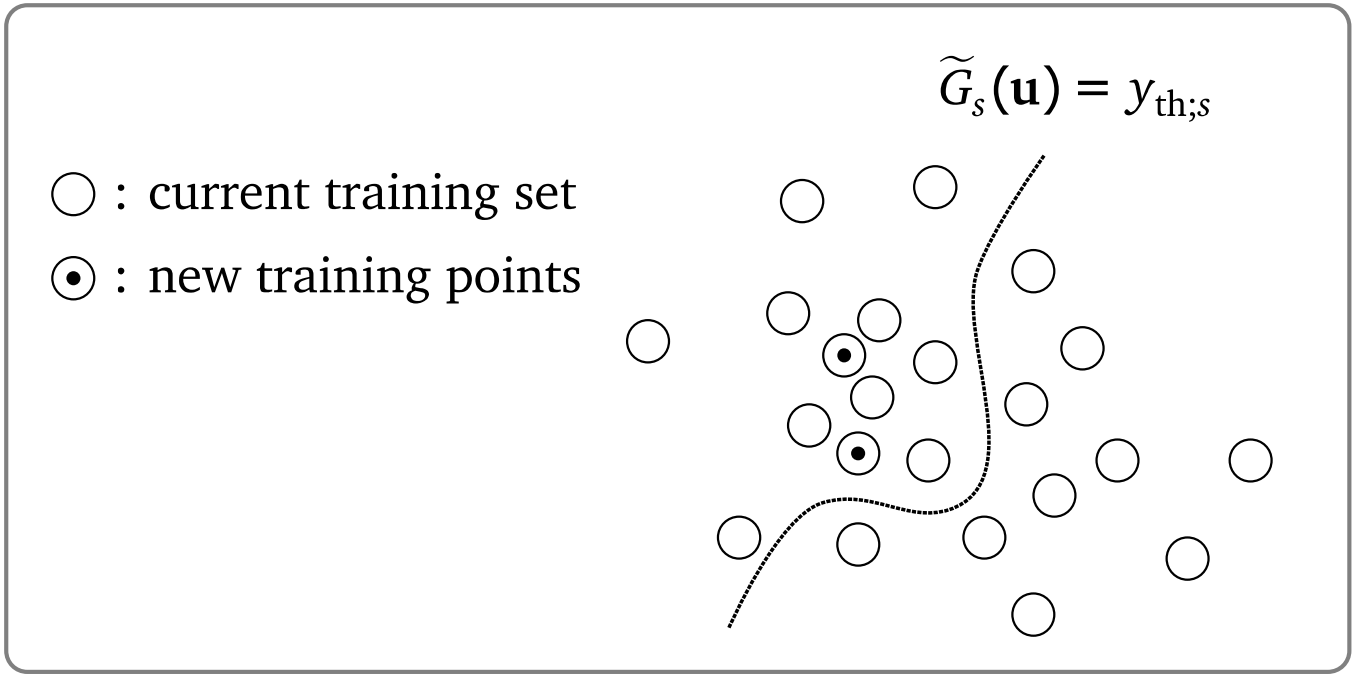
ASVR method (2)



$\mathbf{u}_{\text{SS}}$ samples, new training points

Sampling from modified Metropolis algorithm (Au & Beck 2001)

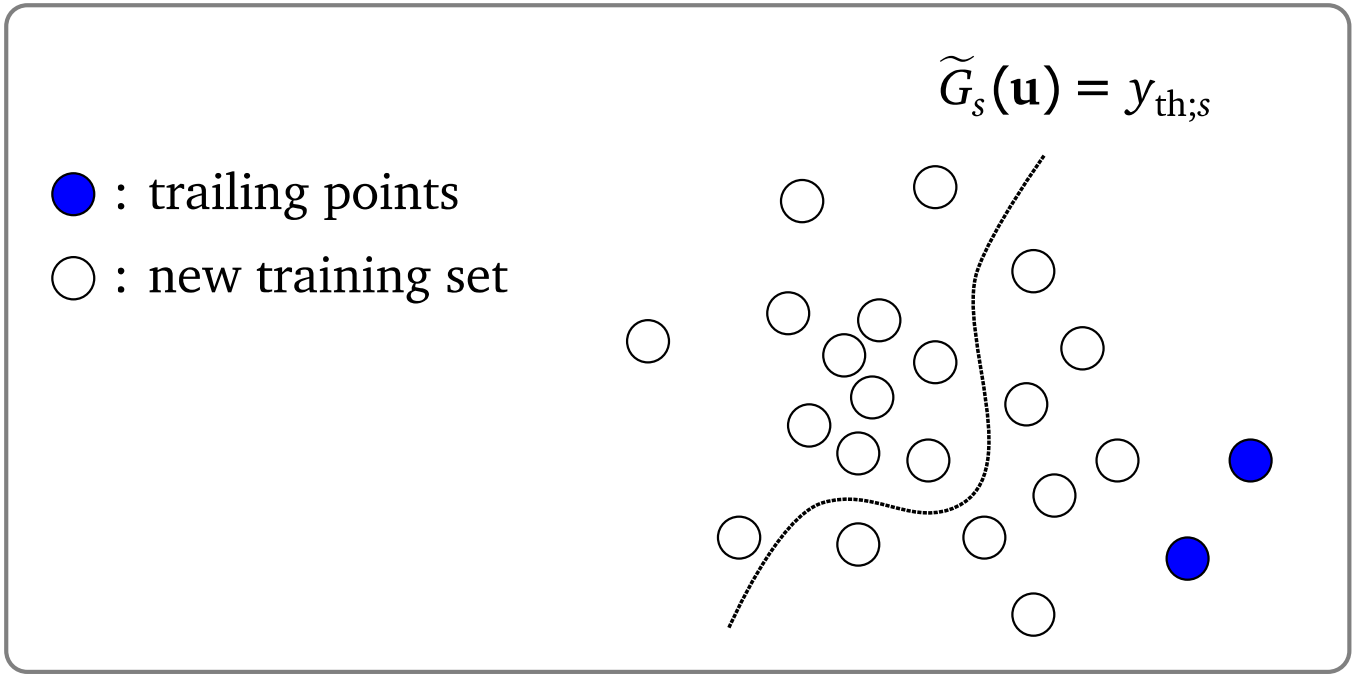
ASVR method (3)



Current training set, new training points

New points added to the current training set

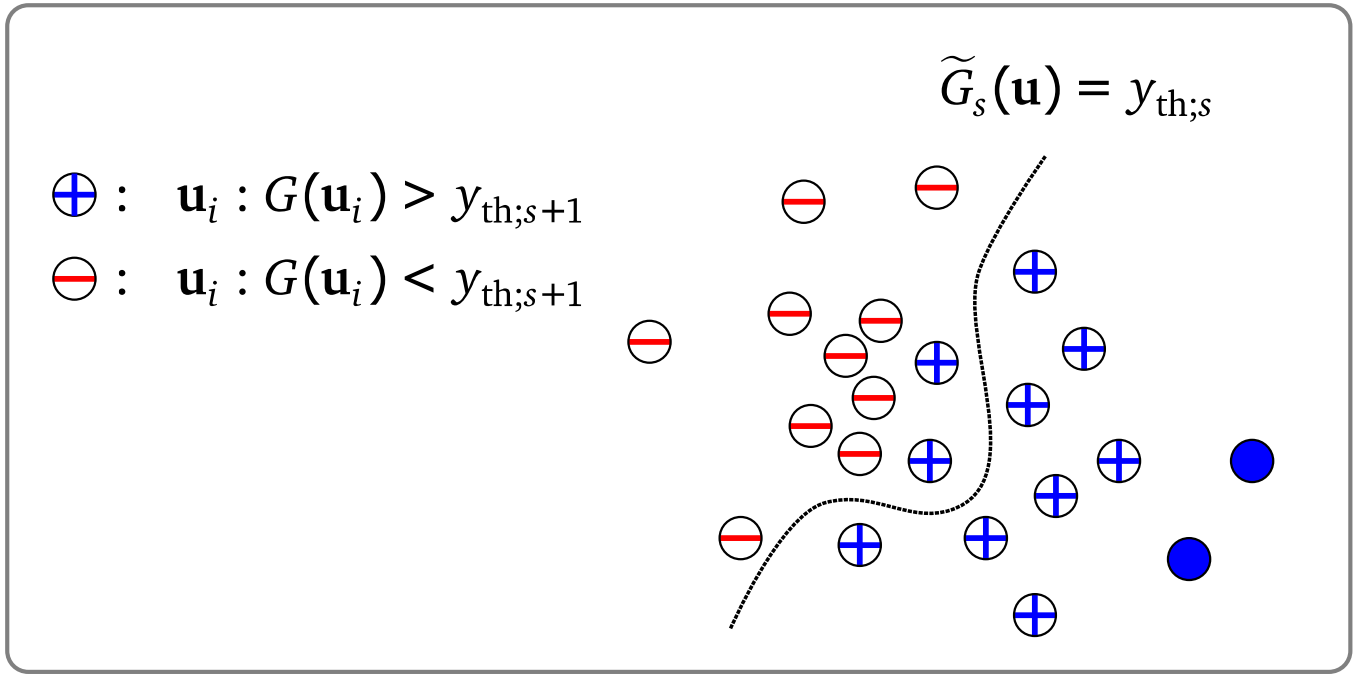
ASVR method (4)



Trailing points, new training set

Points with largest LSF values withdrawn from the current training set

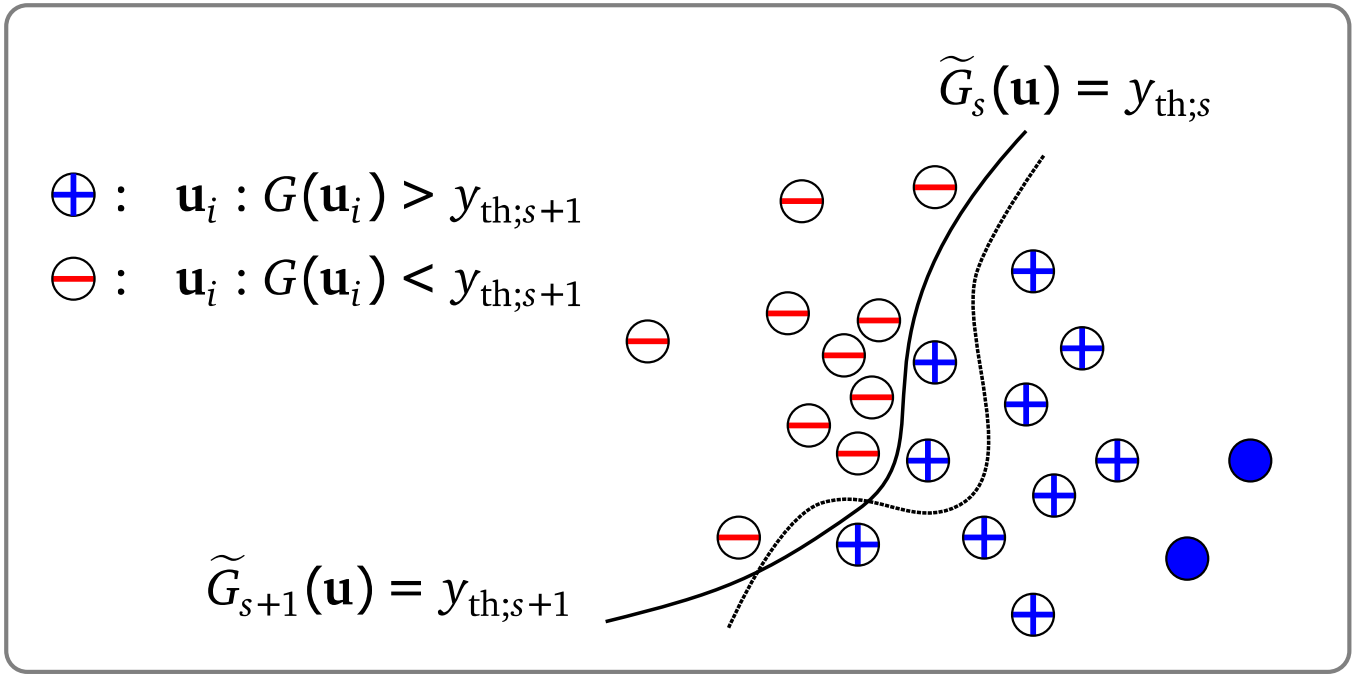
ASVR method (5)



New threshold value $y_{\text{th};s+1}$

Update of intermediate level: $y_{\text{th},\,s+1} = \text{median}\left(\mathbf{g}_{\mathcal{I}_{\text{train}};\,1,\ldots,N}\right)$

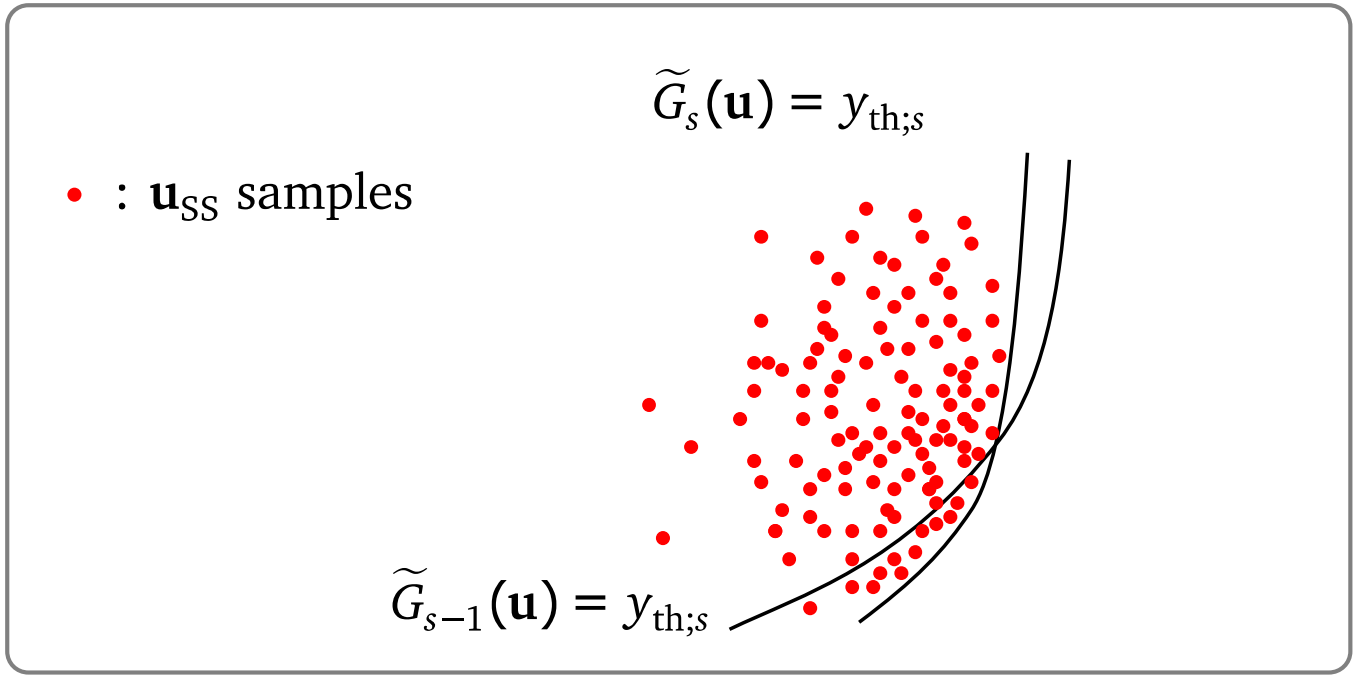
ASVR method (6)



Updated SVR surrogate $\widetilde{G}_{s+1}$

Update of SVR model

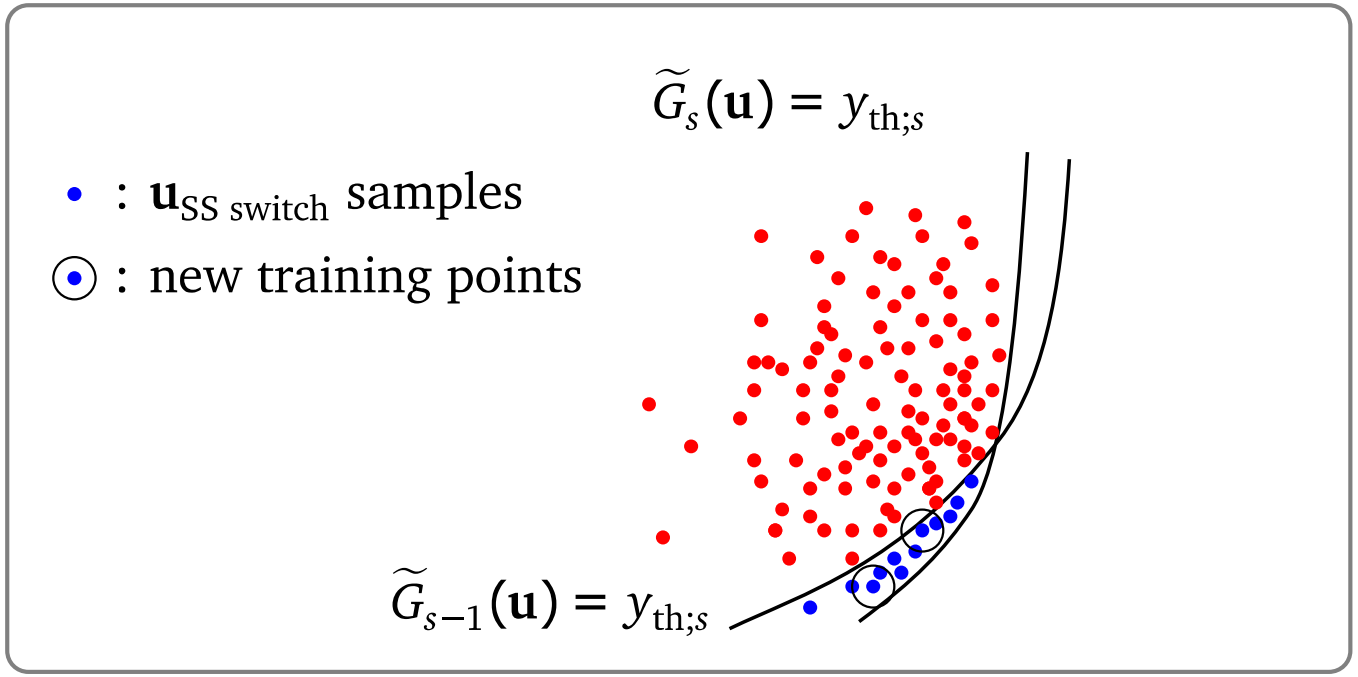
ASVR method (7)



$\mathbf{u}_{\text{SS}}$ samples at current iteration s

Enrichment strategy in the final iterations

ASVR method (8)



$\mathbf{u}_{\text{SS switch}}$ samples, new training points

Enrichment strategy in the final iterations

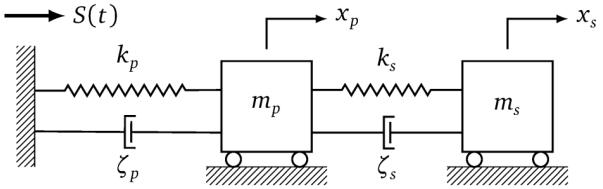
Application examples

Example 1 (from Der Kiureghian & de Stefano 1990)

Limit-state function

$$g(\mathbf{x}) = F_s - 3k_s \sqrt{\frac{\pi S_0}{4\zeta_s\omega_s^3}\left[\frac{\zeta_a\zeta_s}{\zeta_p\zeta_s\left(4\zeta_a^2+\theta^2\right)+\gamma\zeta_a^2}\frac{\left(\zeta_p\omega_p^3+\zeta_s\omega_s^3\right)\omega_p}{4\zeta_a\omega_a^4}\right]}$$

where $\omega_p = \sqrt{k_p/m_p}$, $\omega_s = \sqrt{k_s/m_s}$, $\omega_a = (\omega_p+\omega_s)/2$, $\zeta_a = (\zeta_p+\zeta_s)/2$, $\gamma = m_s/m_p$ and $\theta = (\omega_p-\omega_s)/\omega_a$.



Random variables (8)

Variable	m_p	m_s	k_p	k_s	ζ_p	ζ_s	F_s	S_0
Distribution				Lognormal				
Mean	1.5	0.01	1	0.01	0.05	0.02	[15.0–27.5]	100
c.o.v.	0.1	0.1	0.2	0.2	0.4	0.5	0.1	0.1

Previous results (Bourinet et al. 2011)

	FORM[a] $e_1 = 1\times10^{-3}$ $e_2 = 5\times10^{-3}$	SORM	SS 3×10^5/step $\times500$	^{2}SMART 573/step $\times50$
μ_{F_s}	P_f $N_{\text{call}}(N_{\text{iter}})$	P_f N_{call}	P_f N_{call} (emp. c.o.v.)	P_f N_{call} (emp. c.o.v.)
27.5	3.91×10^{-6} 2727 (303)[b]	3.70×10^{-7} (2727+)44	$\mathbf{3.78\times10^{-7}}$ 21×10^5 (0.040)	3.66×10^{-7} 4011 (0.096)

Reference value in bold characters.

[a] Details on e_1 and e_2 convergence criteria in [22].

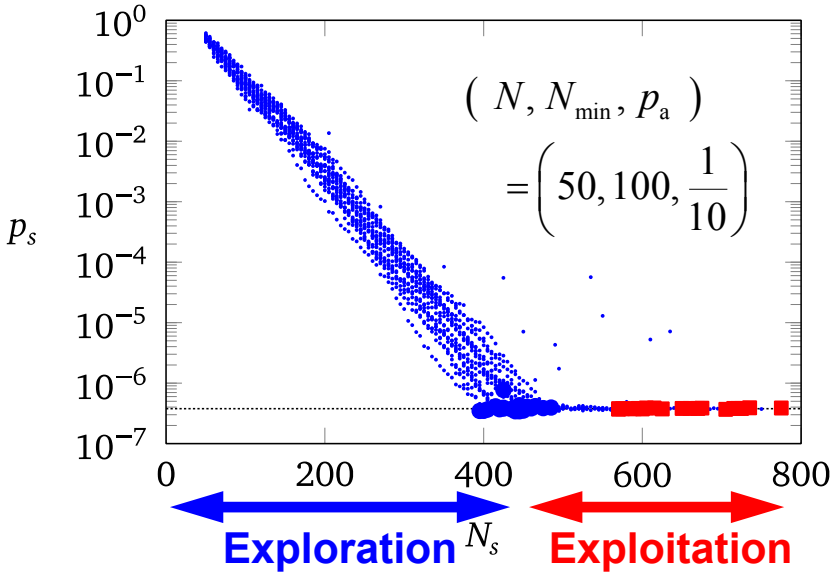
[b] Stepsize = 0.025 (Armijo's rule otherwise).

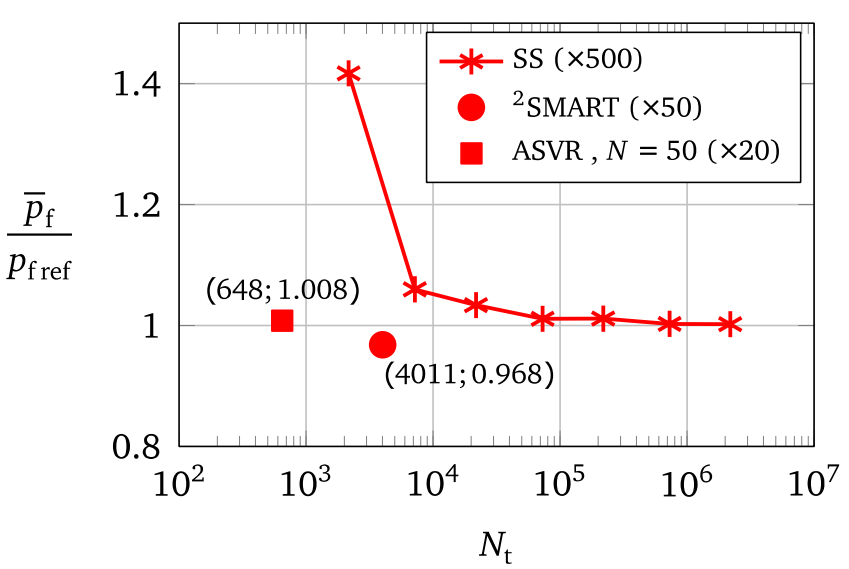
Example 1 (from Der Kiureghian & de Stefano 1990)

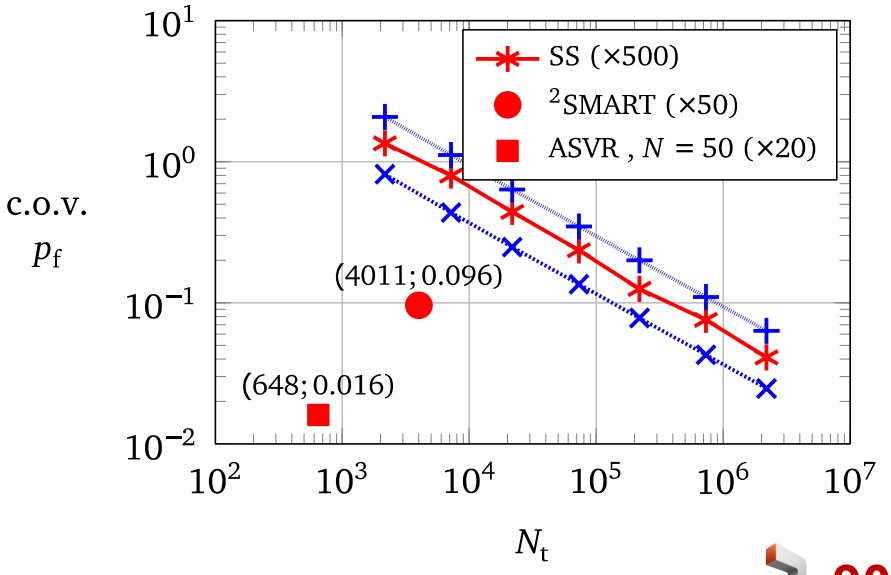
Main characteristics of the reliability assessment problem

- ✓ Rare event: $p_{\mathrm{f\,ref}} = 3.78 \times 10^{-7}$
- ✓ Single MPFP (but very hard to find with FORM!)
- ✓ Highly curved limit-state surface
- ✓ Variable sensitivities at MPFP are different from those at mean

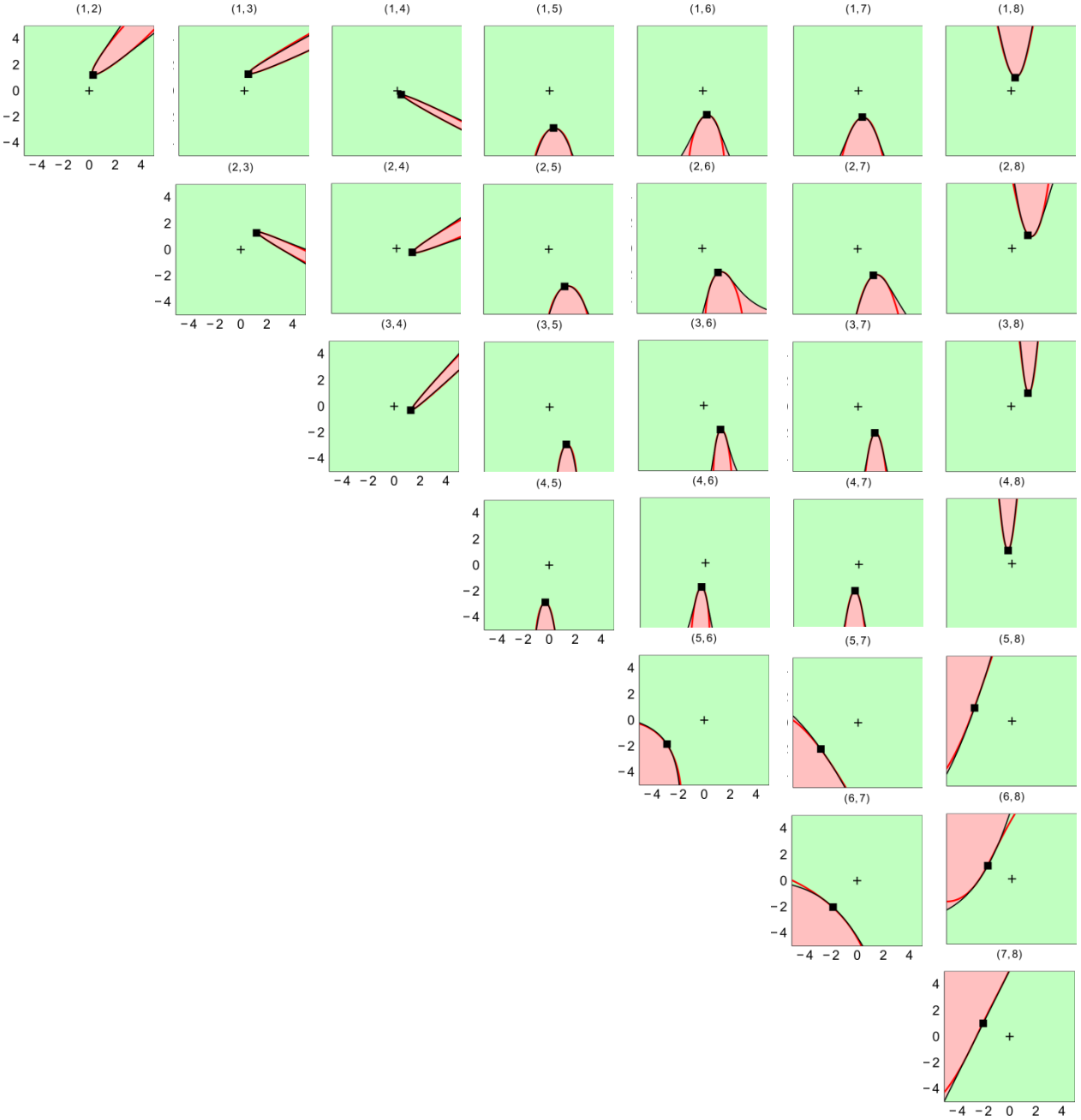
Failure probability estimates







Shape of the limit-state surface



Example 2 (Rackwitz 2001)

- **Limit-state function**

$$g(\mathbf{x}) = g(x_1, \ldots, x_n) = (n + a\sigma\sqrt{n}) - \sum_{i=1}^{n} x_i$$

- **Random variables**
  ($n = 100$, 250)

where X_i, $i = 1, \ldots, n$, are i.i.d. lognormal random variables, with unit means and standard deviations equal to $\sigma = 0.2$. We take here $a = 3$.

- **Previous results (Bourinet et al. 2011)**

	FORM $e_1 = 1 \times 10^{-3}$ $e_2 = 1 \times 10^{-3}$	SORM	SS ×500	^{2}SMART ×20
n	P_f $N_{\text{call}}(N_{\text{iter}})$	P_f N_{call}	P_f N_{call} (emp. c.o.v.)	P_f N_{call} (emp. c.o.v.)
100	4.20×10^{-5} 315 (3)	2.97×10^{-3} (315+)5150	100,000/step $\mathbf{1.73 \times 10^{-3}}$ 300,000 (0.023)	2012/step 1.74×10^{-3} 6036 (0.022)
250	2.82×10^{-6} 765 (3)	5.77×10^{-3} (765+)31,625	100,000/step $\mathbf{1.59 \times 10^{-3}}$ 300,000 (0.023)	3569/step 1.61×10^{-3} 10,707 (0.025)

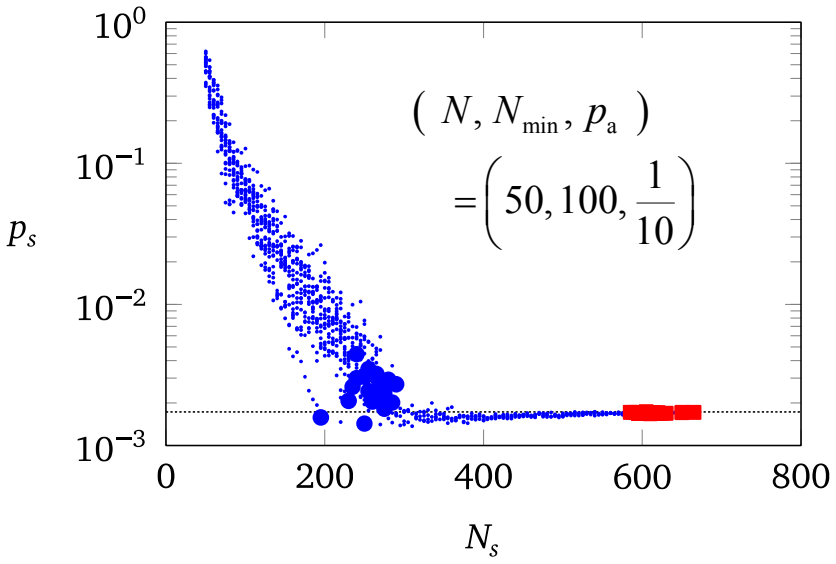
Reference value in bold characters.

Example 2 (Rackwitz 2001)

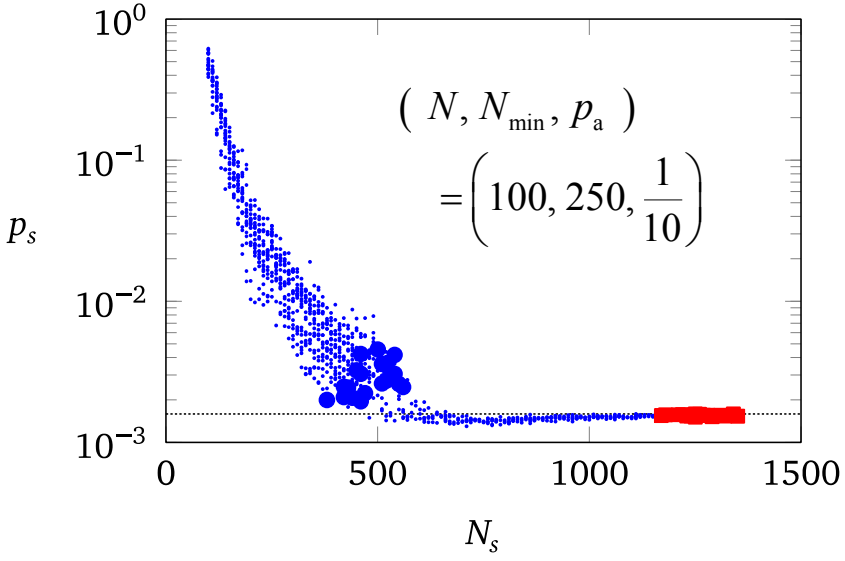
Main characteristics of the reliability assessment problem

- ✓ $p_{\mathrm{f\,ref}} = 1.73 \times 10^{-3}$ ($n = 100$) , $p_{\mathrm{f\,ref}} = 1.59 \times 10^{-3}$ ($n = 250$)
- ✓ Single MPFP
- ✓ High dimension ($n = 100$ and $n = 250$)
- ✓ Smooth limit-state surface
- ✓ All variables of equal importance

Failure probability estimates

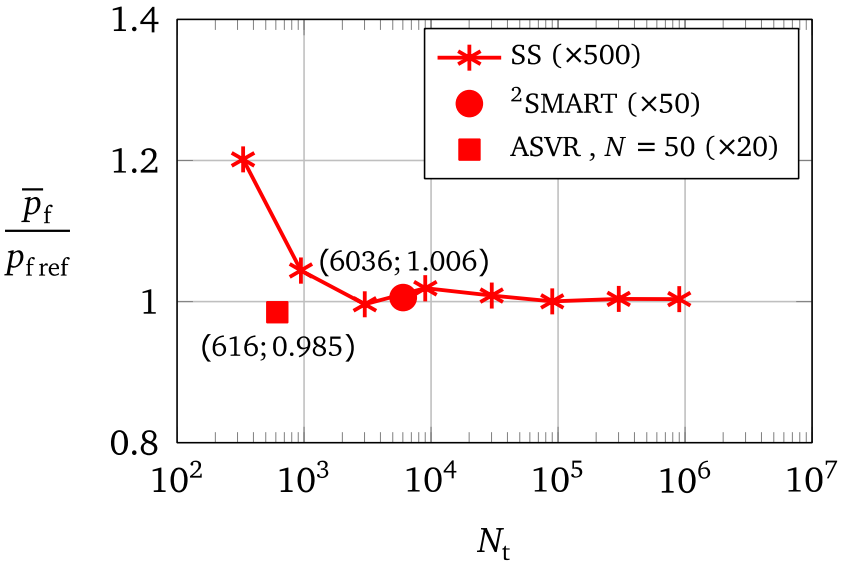


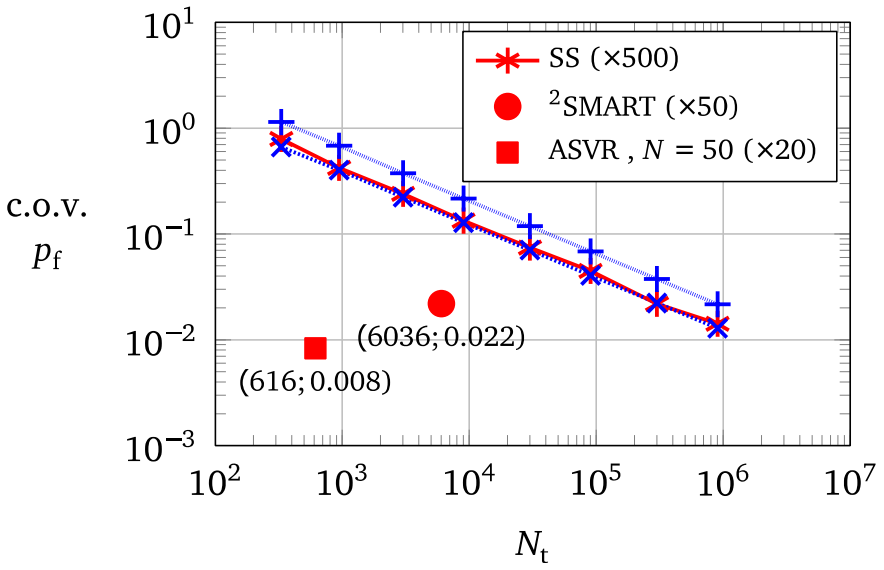
$n = 100$

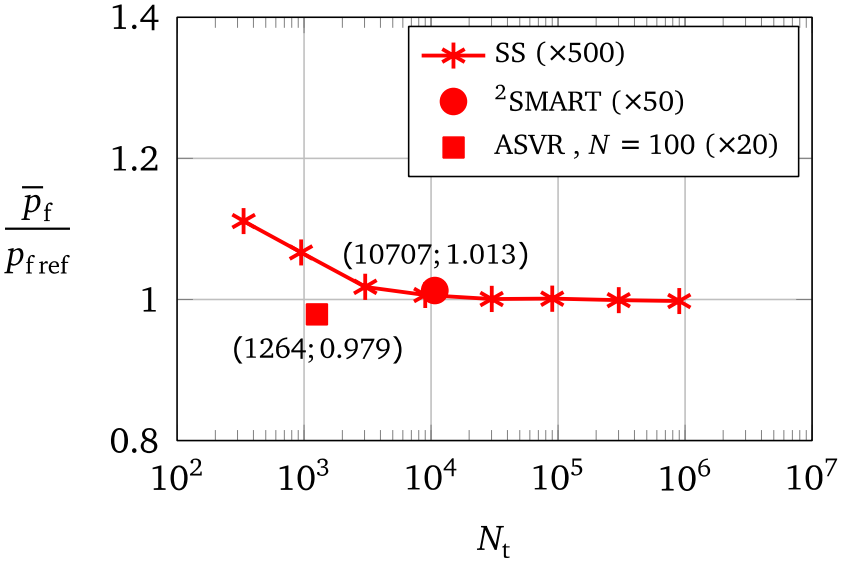


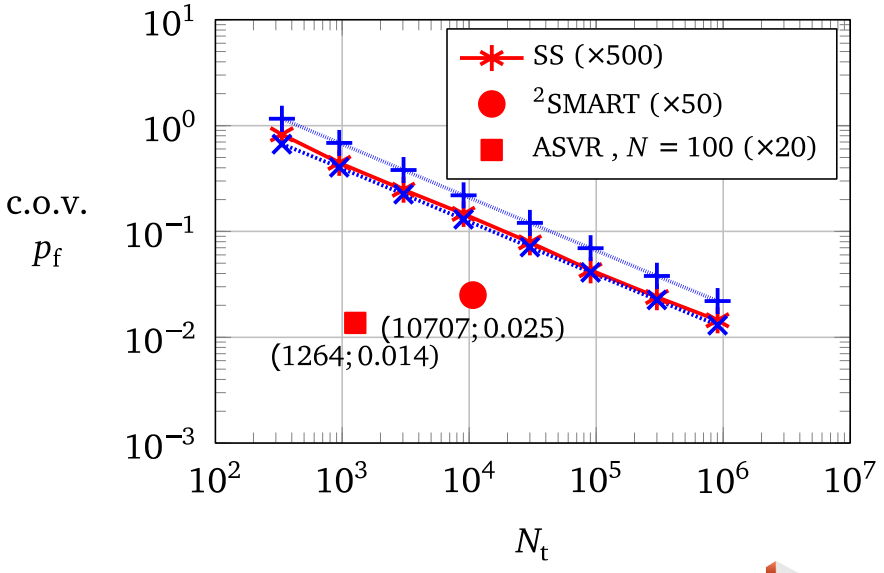
$n = 250$

Failure probability estimates



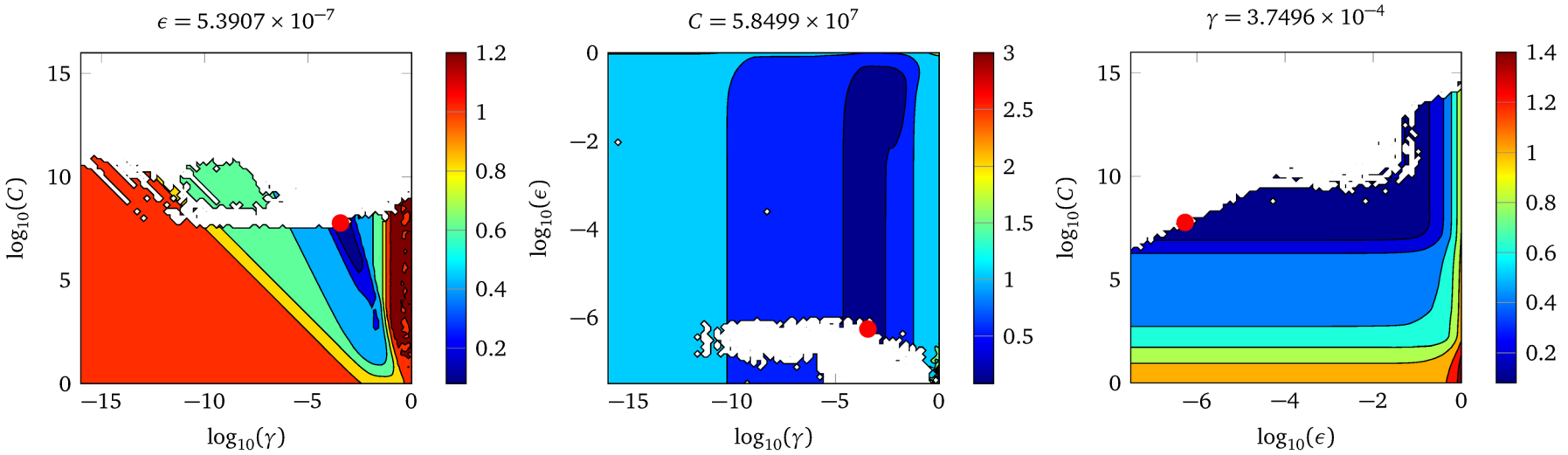




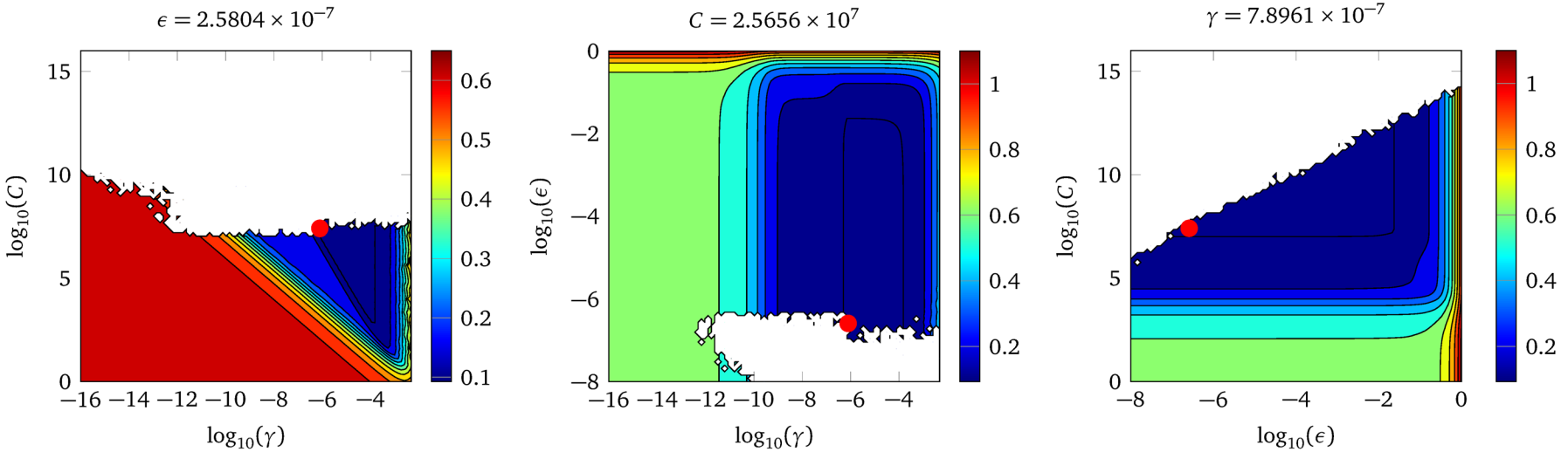


Span approximation of the LOO error ($N = 50$)

Example 1



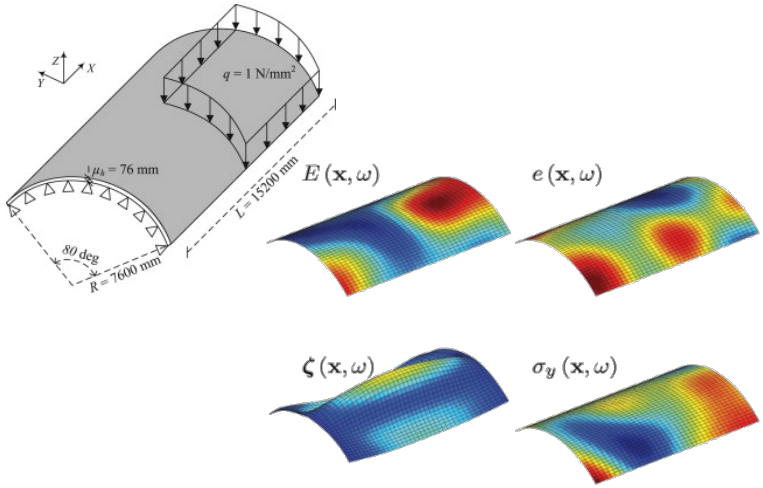
Example 2

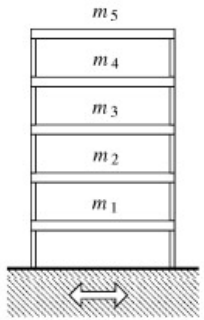


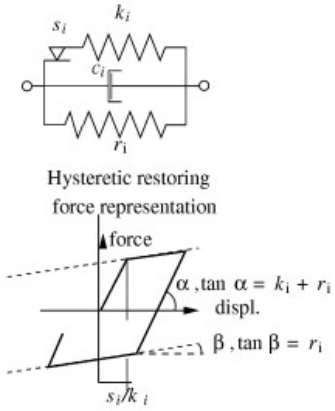
Some perspectives

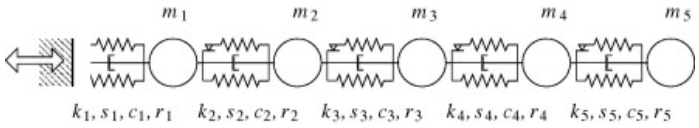
Ongoing works

- ✓ Multiple MPFPs
- ✓ Moderately high dimensional problems (up to about 100-250 r.v.s) with random fields / random processes









In the long run

- ✓ Bias correction with sampling
- ✓ Improve some computational aspects (parallelization of training, efficiency of QP solvers)

Some concluding remarks and personal thoughts

SVM

- ✓ Prefer regression to classification
- ✓ Interior point methods, avoid SMO solvers
- ✓ Very accurate hyperparameter selection (LOO span approximation, stochastic search of optimal values)
- ✓ Kernel: Gaussian RBF, isotropic but also anisotropic
- ✓ Numerical aspects: ill-conditioned Gram matrices, parallel training

More generally (kernel methods)

- ✓ Regularization: important for small training sets
- ✓ Kernels: isotropic vs. anisotropic, stationary vs. nonstationary
- ✓ Trend in kriging / unregularized function basis in SVM (risk of overfitting with small training sets)
- ✓ SVMs vs. kriging: the loss function, closeness of results between kriging and LS-SVR

Thank you for your attention