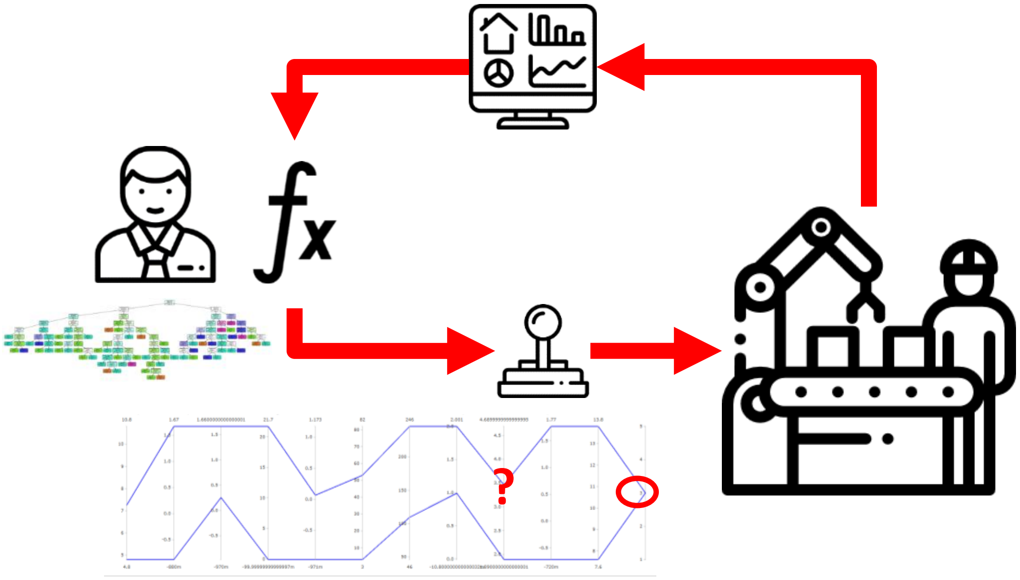
How to manage the impact of measurement uncertainties in manufacturing data on a classification technique?

Big Data & Robust Design



KIMP DM – 2022/23 – Wahb ZOUHRI

Arts et Métiers Institute of Technology

LCFC Conception Fabrication Commande

Plan

Learning outcomes:

- **How to evaluate and handle the impact of noise (uncertainties) in manufacturing data on the performance of classification techniques [SVM as a case study]?**

Keywords : *Support Vector Machine, Measurment uncertainties, Genetic algorithm, Monte-Carlo simulation, Sobol sensitivity analysis, Robust optimisation.*

1st issue :

- **Objective :** How to **optimise** the prediction accuracy of SVM ?

  ⟶ Introduction to SVM.

  ⟶ A genetic algorithm for SVM prediction accuracy optimisation.

2nd issue :

- **Objective :** How to **evaluate** the impact of uncertainties on SVM accuracy ?

  ⟶ Monte-Carlo simulation for uncertainties impact quantification.

  ⟶ Sobol sensitivity analysis for key uncertainties identification.

3rd issue :

- **Objective :** How to **mitigate** the impact of uncertainties on SVM accuracy ?

  ⟶ Bibliographical study.

cc BY NC ND

Plan

Learning outcomes:

- **How to evaluate and handle the impact of noise (uncertainties) in manufacturing data on the performance of classification techniques [SVM as a case study]?**

Keywords : *Support Vector Machine, Measurment uncertainties, Genetic algorithm, Monte-Carlo simulation, Sobol sensitivity analysis, Robust optimisation.*

1st issue :

- **Objective :** How to **optimise** the prediction accuracy of SVM ?

  ⟶ Introduction to SVM.

  ⟶ A genetic algorithm for SVM prediction accuracy optimisation.

2nd issue :

- **Objective :** How to **evaluate** the impact of uncertainties on SVM accuracy ?

  ⟶ Monte-Carlo simulation for uncertainties impact quantification.

  ⟶ Sobol sensitivity analysis for key uncertainties identification.

3rd issue :

- **Objective :** How to **mitigate** the impact of uncertainties on SVM accuracy ?

  ⟶ Bibliographical study.

cc BY NC ND

Plan

1st issue :

- **To do :**
  - A python code: implementation and optimization of SVM using GA.
  - A report briefly explaining SVM and its uses, the issue of Hyperparameters Optimization (*also known as Hyperparameters Tuning*) and how you approached this issue using GA (the length of the report should be between 1000 and 1500 words. You are encouraged to add tables and figures if needed).

2nd issue :

- **To do :**
  - A python code: implementation of Monte-Carlo simulation and Sobol sensitivity analysis for SVM robustness assessment.
  - A report briefly explaining and discussing the approach you implemented (the length of the report should not exceed 1000 words. You are encouraged to add tables and figures if needed).

3rd issue :

- **To do :**
  - A literature report discussing and explaining one or more approaches for mitigating the impact of measurement uncertainties on SVM performance. You can either propose an idea or discuss an approach that already exists in the literature (the length of the report should not exceed 1000 words. You are encouraged to add tables and figures if needed).

cc BY NC ND

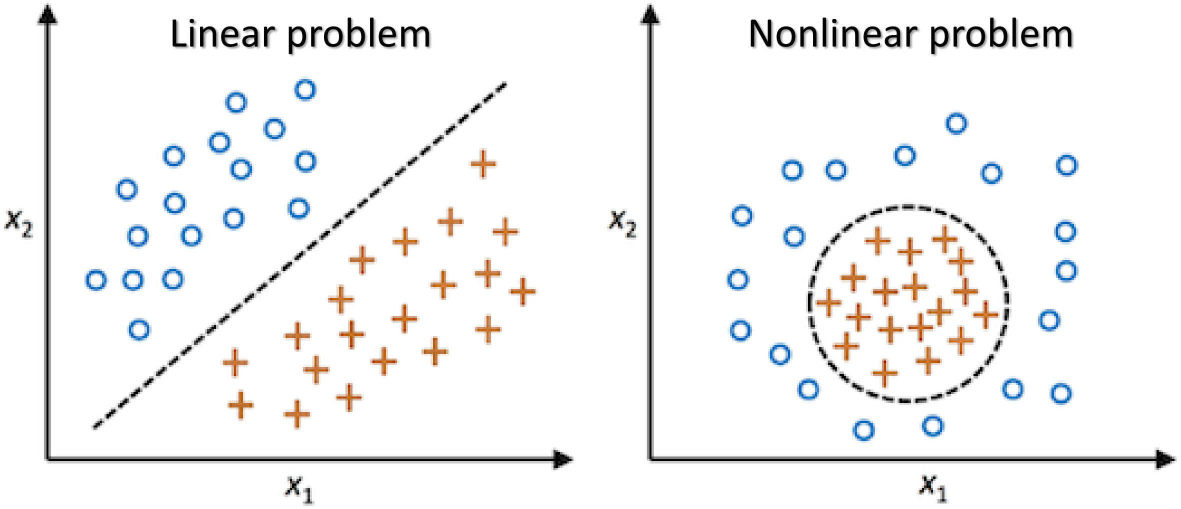
SVM

Introduction

The best way to introduce support vector machines (SVMs) is to consider the simple task of binary classification:

- the exchange rate of a currency will move **up** or **down** based on economic data.
- the decision to **grant** or **deny** a loan to a customer based on personal financial information.
- Predicting product **conformity** or **non-conformity** based on operating parameters.

Being a supervised learning algorithm, the focus of SVM is to learn from a dataset and try to **generalize** and make correct predictions on new data

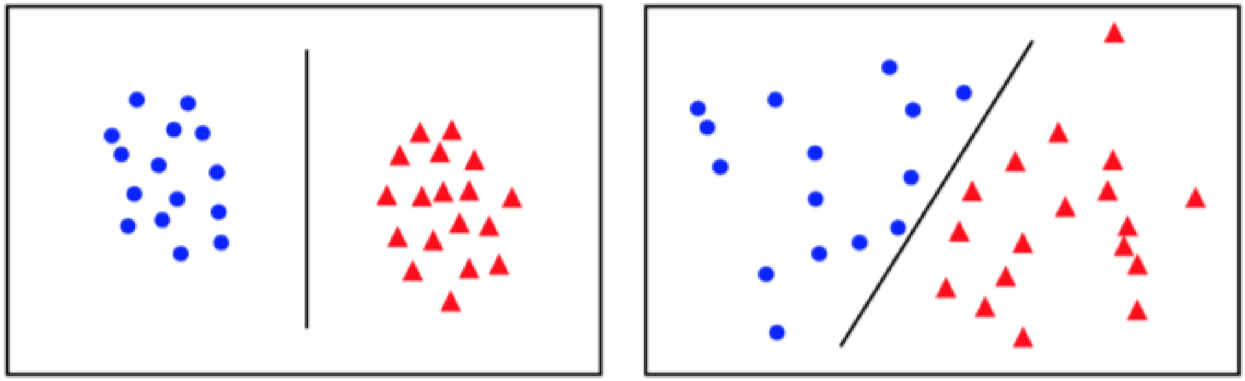


SVM

Linear SVM

Let $\boldsymbol{B_a} = \{\{\boldsymbol{x_n}, \boldsymbol{y_n}\} \in \mathbb{R}^{\boldsymbol{d}} \text{ x } \{\boldsymbol{-1}, \boldsymbol{1}\}\}_{\boldsymbol{n=1\ldots N}}$ be a set of labeled learning examples.

The points $\{\boldsymbol{x_n}, \boldsymbol{y_n}\}$ are linearly separable if there is a hyperplane that correctly separates the two classes' data.



The objective of SVM is to learn a decision function $\boldsymbol{f(x)}$, such that:

$$\boldsymbol{f(x_n)} \begin{cases} \boldsymbol{> 0} & \boldsymbol{si} \quad \boldsymbol{y_n = \ \ 1} \\ \boldsymbol{< 0} & \boldsymbol{si} \quad \boldsymbol{y_n = -1} \end{cases}$$

equivalently $\boldsymbol{y_n f(x_n) > 0}$ for any correctly predicted example.

Linear SVM

Decision function

The expression for the decision function is :

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$

- **w** is the normal vector to the hyperplane, and **b** the bias
- **w** is known as the weight vector.

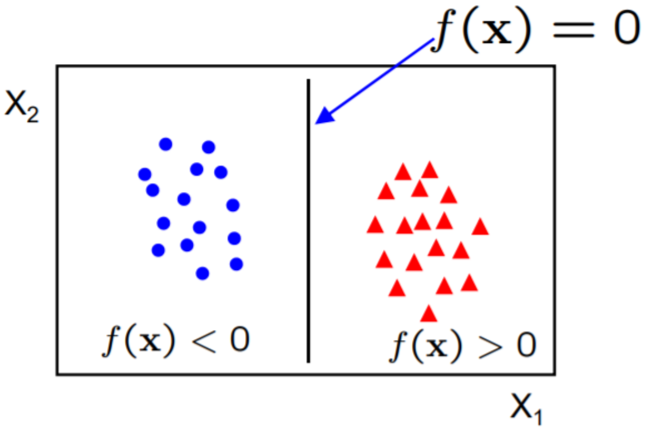
Linear SVM

Decision function

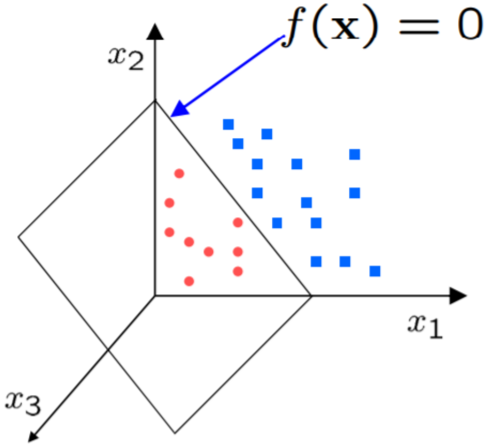
The expression for the decision function is :

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$

- **w** is the normal vector to the hyperplane, and **b** the bias
- **w** is known as the weight vector.



Hyperplane in 2D = Line



Hyperplane in 3D = Plan

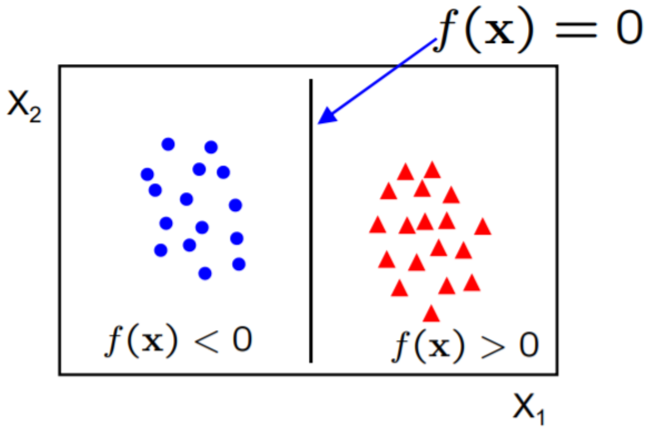
Linear SVM

Decision function

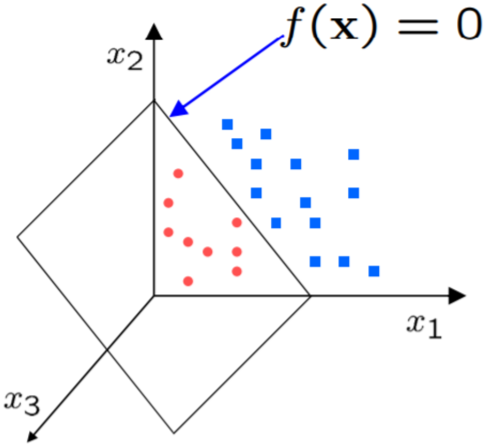
The expression for the decision function is :

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$

- **w** is the normal vector to the hyperplane, and **b** the bias
- **w** is known as the weight vector.



Hyperplane in 2D = Line



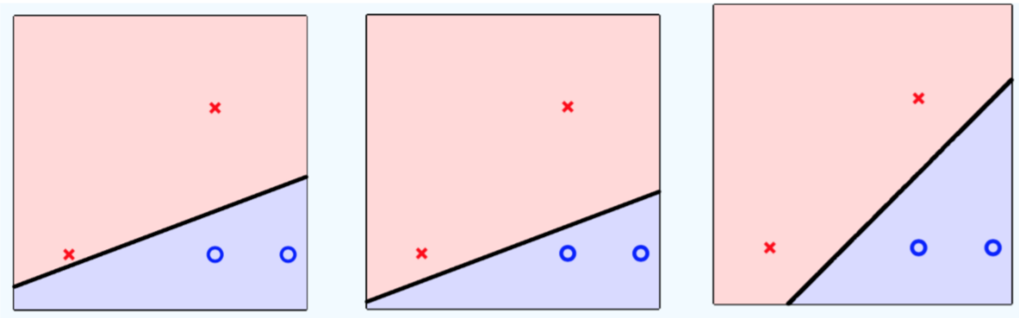
Hyperplane in 3D = Plan

The hyperplane is the set of points x_n such that : $\mathbf{w}^{\mathbf{T}}\mathbf{x_n} + \mathbf{b} = \mathbf{0}$

Linear SVM

Separating hyperplane

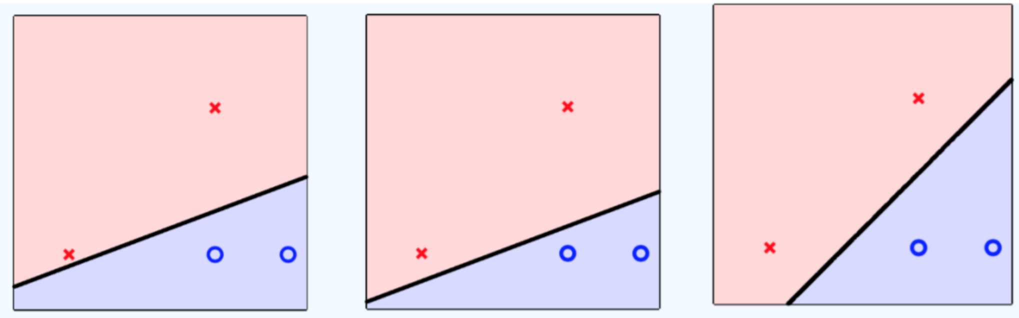
There are an infinite number of lines to separate data that are linearly separable.



Linear SVM

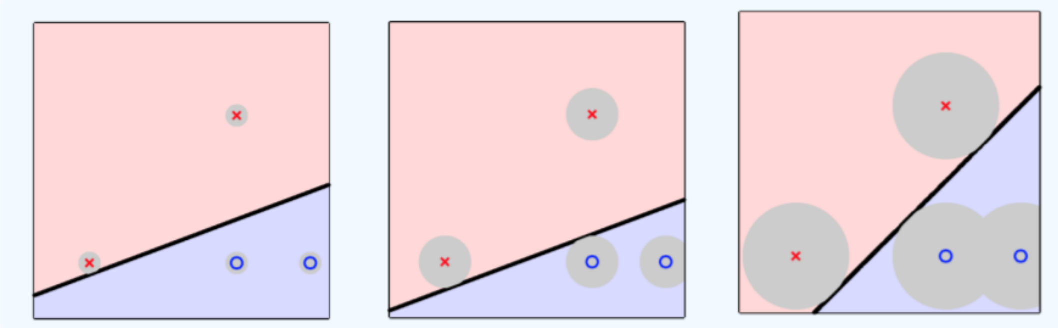
Separating hyperplane

There are an infinite number of lines to separate data that are linearly separable.



Which one to choose? :

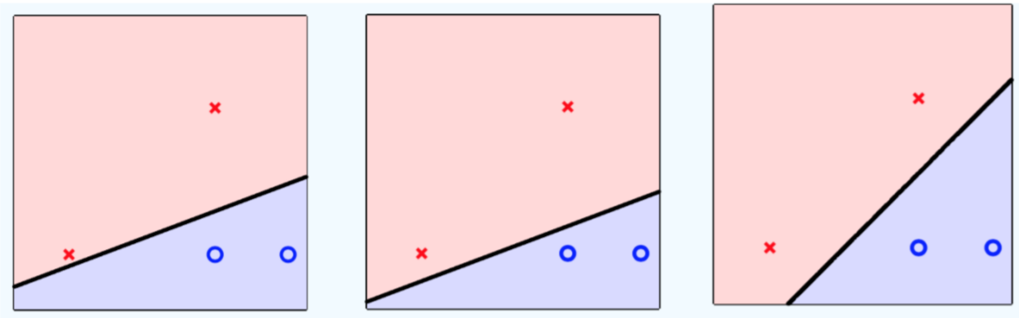
The SVM chooses the hyperplane that most tolerates the noise in the training data



Linear SVM

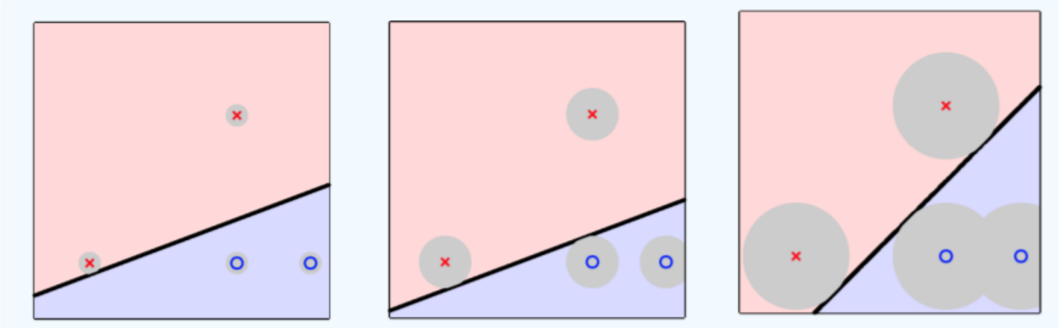
Separating hyperplane

There are an infinite number of lines to separate data that are linearly separable.

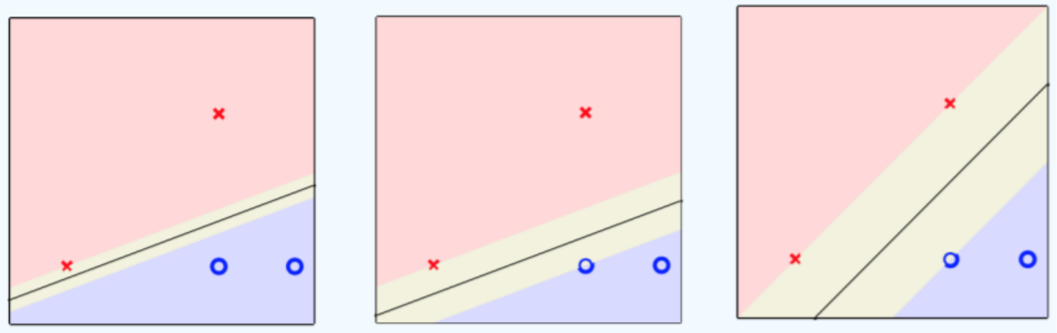


Which one to choose? :

The SVM chooses the hyperplane that most tolerates the noise in the training data

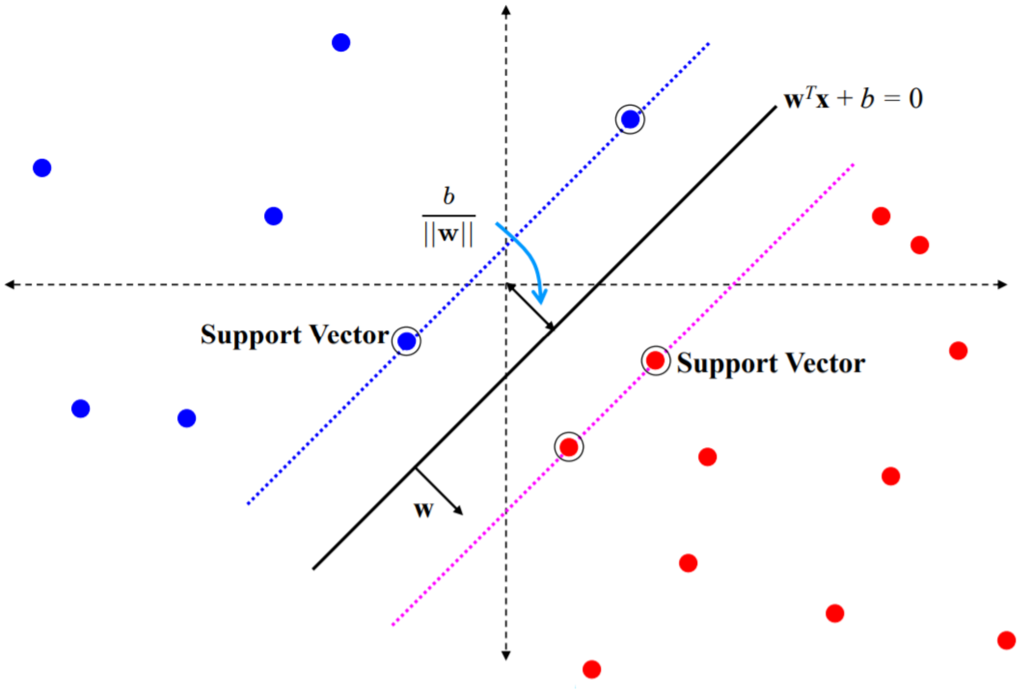


This results in a maximal margin between the two classes.



Linear SVM

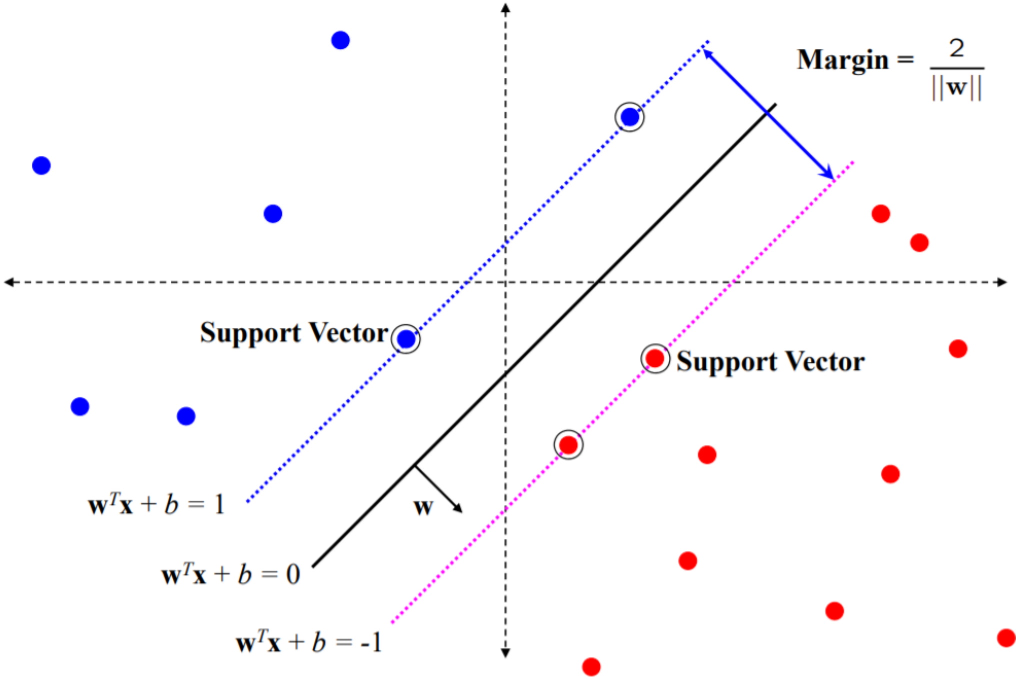
Summary of SVM concepts



- The decision function is defined by : $\mathbf{f(x_n)} = \mathbf{w^T x_n + b}$
- The hyperplane is the set of points $\{\mathbf{x_n}\}$ that satisfy : $\mathbf{w^T x_n + b} = 0$
- Support vectors are the closest data points to the hyperplane. They influence the position and orientation of the hyperplane.

Linear SVM

Summary of SVM concepts


Margin = $\frac{2}{||\mathbf{w}||}$
Support Vector
Support Vector
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$
w

- The support vectors define two hyperplanes $\{H^+, H^-\}$.
- These support vectors satisfy the following equalities :
  - For the set of $\mathbf{x}_+ \in \mathbf{H}^+ \mathbf{: w^T x_+ + b = 1}$
  - For the set of $\mathbf{x}_- \in \mathbf{H}^- \mathbf{: w^T x_- + b = \text{-}1}$
- The margin is therefore defined as the distance between $H^+$and H^- along the unit vector $\frac{\vec{w}}{\|w\|}$:

$$Margin = \frac{2}{\|w\|}$$

Linear SVM

Linear SVM formulation

The objective of SVM is to maximize the margin; this can be formulated as an optimization problem :

$$\begin{array}{lll} max & \frac{1}{\|w\|} & \\ s.t & y_i(w^T.x_i + b) \geq 1 & \forall\, i = 1, \dots, N \\ & y_i \;\in \{-1,1\} & \\ & w \in\; \mathbb{R}^d; b \in \mathbb{R} & \end{array}$$

The quadratic formulation equivalent to this formulation :

$$\begin{array}{lll} min & \frac{1}{2}w^T.w & \\ s.t & y_i(w^T.x_i + b) \geq 1 & \forall\, i = 1, \dots, N \\ & y_i \;\in \{-1,1\} & \\ & w \in\; \mathbb{R}^d; b \in \mathbb{R} & \end{array}$$

This is a quadratic optimization problem subject to linear constraints, so **there is a unique minimum** (a unique vector w).

Linear SVM

Formulation de SVM linéaire

The quadratic formulation equivalent to this formulation :

$$\begin{aligned} &min && \tfrac{1}{2} w^T . w \\ &s.t && y_i (w^T . x_i + b) \geq 1 \qquad \forall\, i = 1, \dots, N \\ & && y_i \in \{-1,1\} \\ & && w \in \mathbb{R}^d ; b \in \mathbb{R} \end{aligned}$$

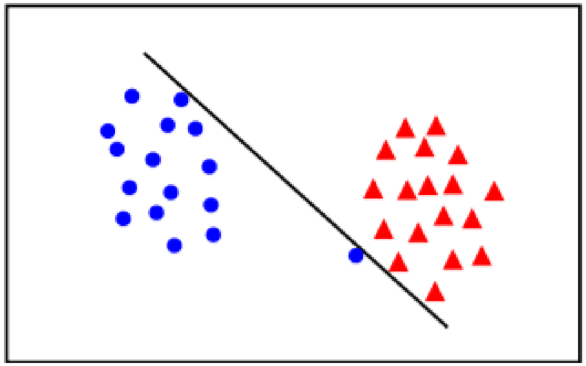
With the help of programming languages (e.g. python) we could solve this optimization problem and find the values of **w** and **b**.

The prediction of a new data $\mathbf{x_n}$ is done by evaluating the sign of the decision function $\mathbf{f}(\mathbf{x_n})$:

- IF $\mathbf{f}(\mathbf{x_n}) = \mathbf{w^T x_n} + \mathbf{b} < 0$ → negative class ($y_\mathbf{n} = -1$)
- IF $\mathbf{f}(\mathbf{x_n}) = \mathbf{w^T x_n} + \mathbf{b} > 0$ → positive class ($y_\mathbf{n} = +1$)

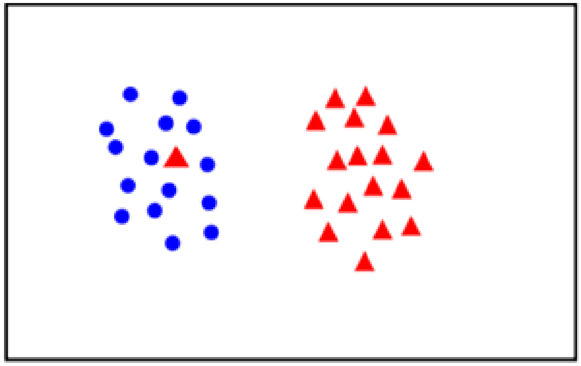
Linear SVM

Limitations of linear SVM



The points can be separated linearly, but the margin is very narrow.

A very noise-sensitive model!

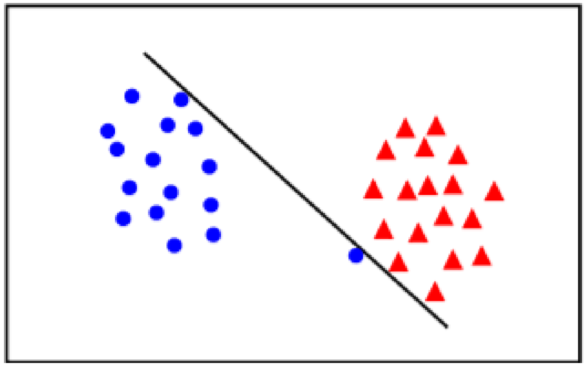


SVM fails to find a model that perfectly separates the data.

Algorithm does not converge!

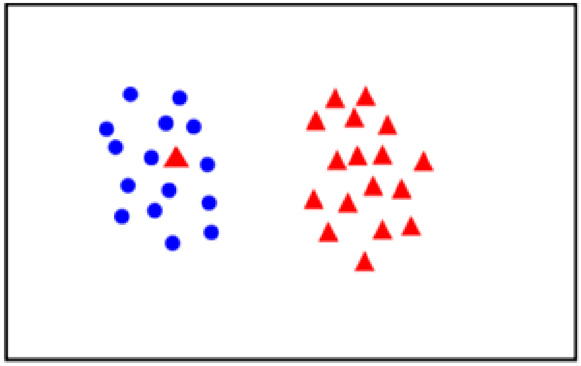
Linear SVM

Limitations of linear SVM



The points can be separated linearly, but the margin is very narrow.

A very noise-sensitive model!



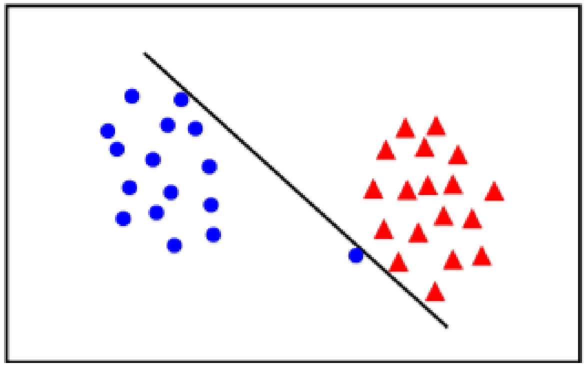
SVM fails to find a model that perfectly separates the data.

Algorithm does not converge!

What if we accept that some examples are on the wrong side of the hyperplane?

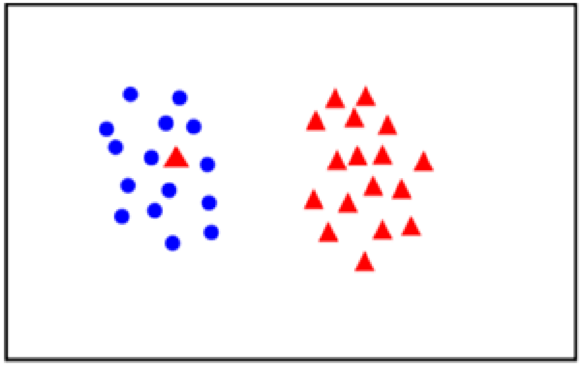
Linear SVM

Limitations of linear SVM



The points can be separated linearly, but the margin is very narrow.

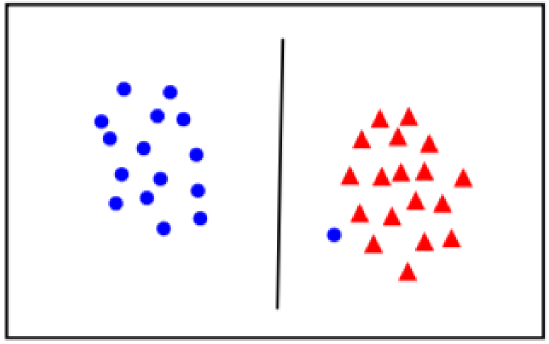
A very noise-sensitive model!

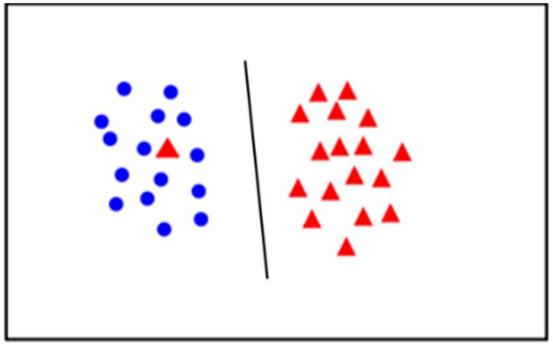


SVM fails to find a model that perfectly separates the data.

Algorithm does not converge!

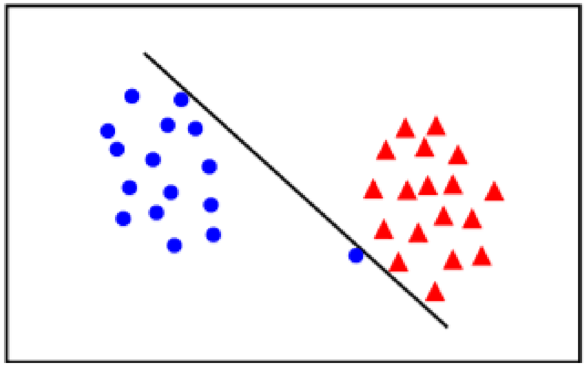
What if we accept that some examples are on the wrong side of the hyperplane?





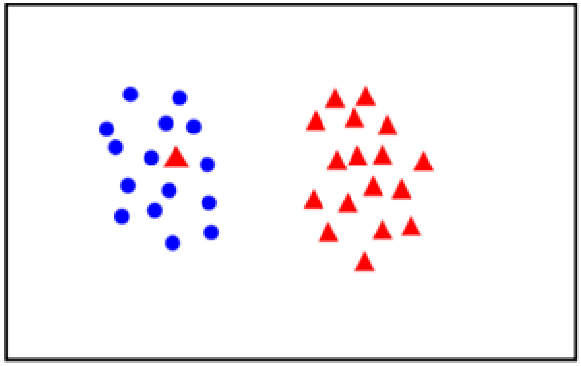
Linear SVM

Limitations of linear SVM



The points can be separated linearly, but the margin is very narrow.

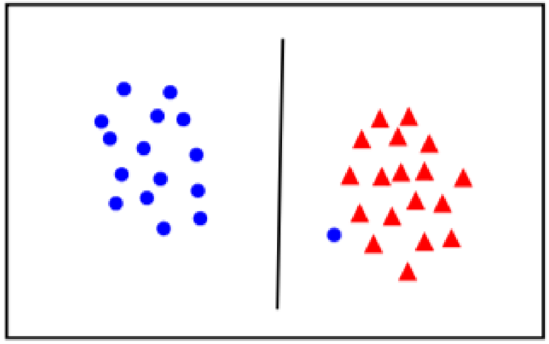
A very noise-sensitive model!

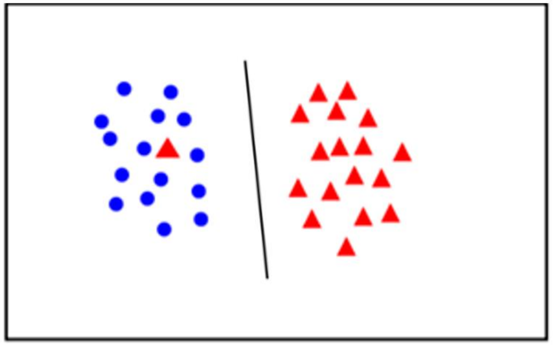


SVM fails to find a model that perfectly separates the data.

Algorithm does not converge!

What if we accept that some examples are on the wrong side of the hyperplane?





Better generalization of the model on new data.

Linear SVM

Soft margin

For some non-linearly separable datasets, a linear separator would still give good results.

Condition: accept that some examples do not respect the constraint.

Problem: the current formulation of SVM does not allow this (hard margin).

Linear SVM

Soft margin

For some non-linearly separable datasets, a linear separator would still give good results.

Condition: accept that some examples do not respect the constraint.

Problem: the current formulation of SVM does not allow this (hard margin).

Solution :

- Relax the constraints :
  - accept that the margin is crossed.
  - accept that some examples are on the wrong side of the hyperplane.
- Penalize these relaxations in the function to be optimized.

Linear SVM

Soft margin

To penalize slacking, a slack variable $\boldsymbol{\xi_n}$ is associated with each learning example $(\boldsymbol{x_n}, \boldsymbol{y_n})$.

New constraint: :

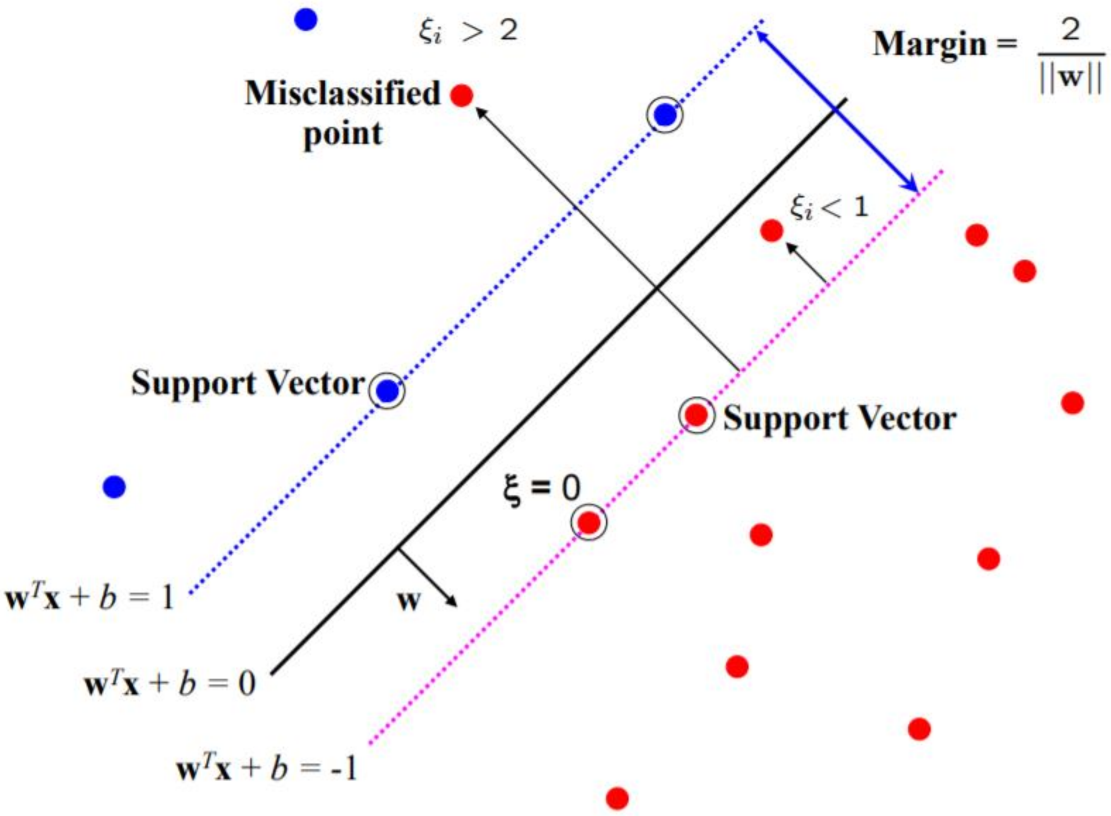
$$\boldsymbol{y_n}\left(\mathbf{w}^{\mathbf{T}}\mathbf{x_n} + \mathbf{b}\right) > \mathbf{1} - \boldsymbol{\xi_n}$$

ξ_n informs how much an example $(\boldsymbol{x_n}, \boldsymbol{y_n})$ violates the constraint :

- IF $\xi_n = 0$ the example respects the constraint.
- IF $\xi_n > 1$ the example is misclassified.
- IF $0 < \xi_n < 1$ the example is well classified but has crossed the margin.

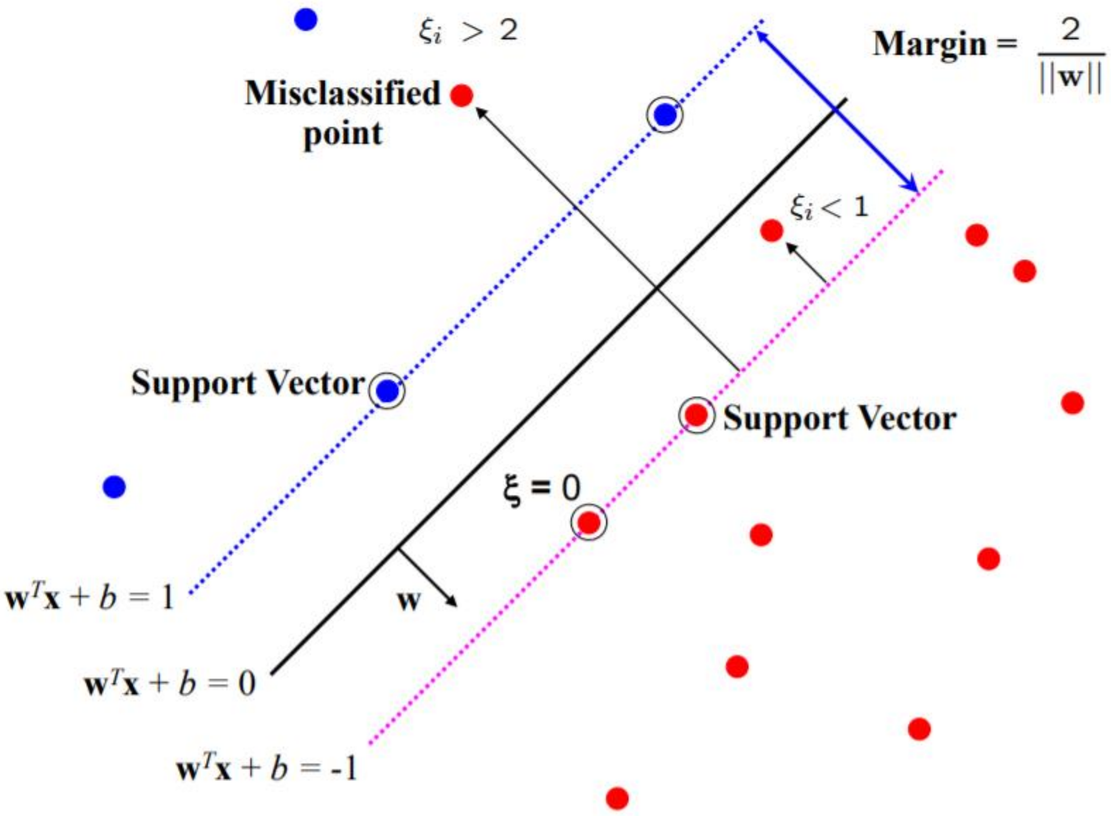
Linear SVM

Soft margin


$\xi_i > 2$
Margin = $\frac{2}{||\mathbf{w}||}$
Misclassified point
$\xi_i < 1$
Support Vector
Support Vector
$\xi = 0$
$\mathbf{w}$
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$

Linear SVM

Soft margin



$\xi_i > 2$
Margin = $\frac{2}{||\mathbf{w}||}$
Misclassified point
$\xi_i < 1$
Support Vector
Support Vector
$\xi = 0$
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$

New objective :

We want $\sum \xi_n$ to be minimal!

Linear SVM

Soft margin

The **new formulation** of soft-margin SVM is as follows :

$$\begin{aligned} & min && \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i && \\ & s.t && y_i(w^T.x_i + b) \geq 1 - \xi_i && \forall\, i = 1, \dots, n \\ & && \xi_i \geq 0 && \forall\, i = 1, \dots, n \end{aligned}$$

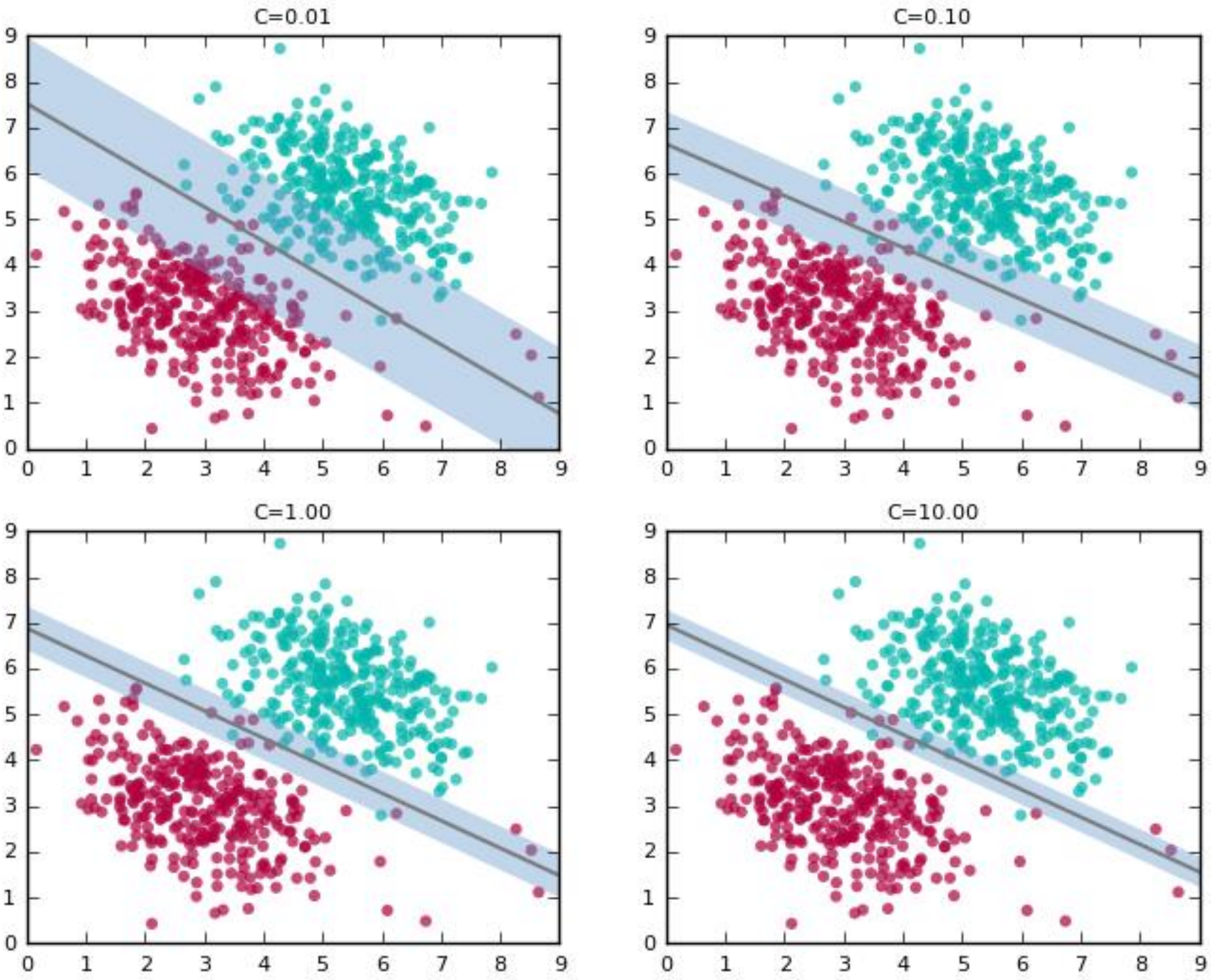
- Each constraint can be satisfied if $\boldsymbol{\xi_n}$ is large enough.
- **C** is a regularization parameter that handles the tradeoff between a maximum margin and constraint relaxation :
  - A small value of **C** allows constraints to be easily ignored → **large margin**
  - A large value of **C** makes constraints hard to ignore → **narrow margin**
- This is still a quadratic optimization problem and there is a unique minimum. Note that there is only one hyperparameter*, C.

* a hyperparameter is a parameter whose value is used to control the learning process. Its value is defined before starting the learning process.

Linear SVM

Soft margin

Illustration of the influence of the hyperparameter C on the width of the margin:



Non-linear SVM

Non-linear separator

Basic idea: if you can't separate positives from negatives in a low-dimensional space using a hyperplane, then project all the data into a higher-dimensional space where you can separate them.

Apply to the data a transformation $\boldsymbol{\phi}: \mathbb{R}^{\boldsymbol{d}} \rightarrow \mathbb{R}^{\tilde{\boldsymbol{d}}}$ to a higher-dimensional space :

$$z_n = \boldsymbol{\phi}(x_n)$$

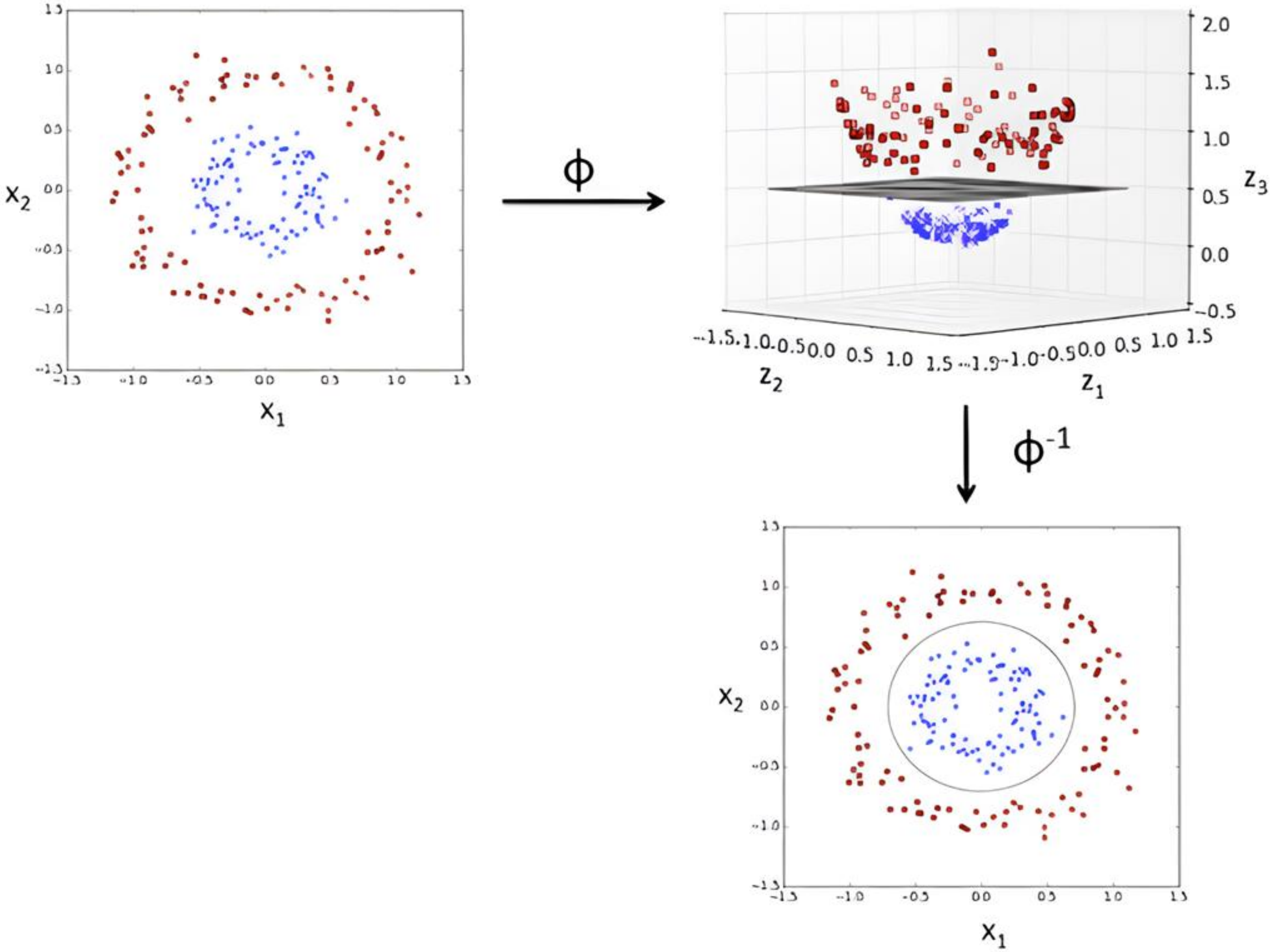
After transforming the data, **we solve the SVM problem in this new redescription space**.

$$\begin{array}{lll} min & \frac{1}{2}\widetilde{w}^T.\widetilde{w} & \\ s.c & y_n(\widetilde{w}^T.z_n + b) \geq 1 & \forall\, n = 1, \dots, N \\ & y_n \in \{-1,1\} & \end{array}$$

Non-linear SVM

Non-linear separator

In this way, we can separate the two classes represented in the new space by a linear hyperplane that becomes a nonlinear decision boundary if we project it back onto the original feature space.



Non-linear SVM

kernel trick

Let's consider the following transformation:

$$\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$\phi(x) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} {x_1}^2 \\ {x_2}^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}$$

Non-linear SVM

kernel trick

Let's consider the following transformation:

$$\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$\phi(x) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} {x_1}^2 \\ {x_2}^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}$$

Scalar product:

$$\phi(x).\phi(x') = \begin{pmatrix} {x_1}^2 & {x_2}^2 & \sqrt{2}x_1x_2 \end{pmatrix} \begin{pmatrix} {x'_1}^2 \\ {x'_2}^2 \\ \sqrt{2}x'_1x'_2 \end{pmatrix}$$

$$= {x_1}^2\,{x'_1}^2 + {x_2}^2\,{x'_2}^2 + 2x_1\,x'_1\,x_2\,x'_2$$

$$= (x_1\,x'_1 + x_2\,x'_2)^2 = (\boldsymbol{x^T}.\boldsymbol{x'})^{\mathbf{2}}$$

Non-linear SVM

kernel trick

Let's consider the following transformation:

$$\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$\phi(x) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} {x_1}^2 \\ {x_2}^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}$$

Scalar product:

$$\phi(x).\phi(x') = \begin{pmatrix} {x_1}^2 & {x_2}^2 & \sqrt{2}x_1x_2 \end{pmatrix} \begin{pmatrix} {x'_1}^2 \\ {x'_2}^2 \\ \sqrt{2}x'_1x'_2 \end{pmatrix}$$

$$= {x_1}^2\,{x'_1}^2 + {x_2}^2\,{x'_2}^2 + 2x_1\,x'_1\,x_2\,x'_2$$

$$= (x_1\,x'_1 + x_2\,x'_2)^2 = (\boldsymbol{x}^{\boldsymbol{T}}.\boldsymbol{x'})^{\mathbf{2}}$$

In this case, the function $(\boldsymbol{x}^{\boldsymbol{T}}.\boldsymbol{x'})^{\mathbf{2}}$ allows us to avoid having to compute the scalar product of two examples explicitly**, saving us a lot of time.**

Non-linear SVM

kernel trick

Let's consider the following transformation:

$$\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$\phi(x) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} {x_1}^2 \\ {x_2}^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}$$

Scalar product:

$$\phi(x).\phi(x') = \begin{pmatrix} {x_1}^2 & {x_2}^2 & \sqrt{2}x_1x_2 \end{pmatrix} \begin{pmatrix} {x'_1}^2 \\ {x'_2}^2 \\ \sqrt{2}x'_1x'_2 \end{pmatrix}$$

$$= {x_1}^2\, {x'_1}^2 + {x_2}^2\, {x'_2}^2 + 2x_1\, x'_1\, x_2\, x'_2$$

$$= (x_1\, x'_1 + x_2\, x'_2)^2 = (\boldsymbol{x}^{\boldsymbol{T}}.\boldsymbol{x'})^{\mathbf{2}}$$

In this case, the function $(\boldsymbol{x}^{\boldsymbol{T}}.\boldsymbol{x'})^{\mathbf{2}}$ allows us to avoid having to compute the scalar product of two examples explicitly**, saving us a lot of time.**

The goal is to take advantage of this trick in the case of SVMs, in order to reduce the computation time in the nonlinear cases.

Non-linear SVM

Kernel trick - dual formulation

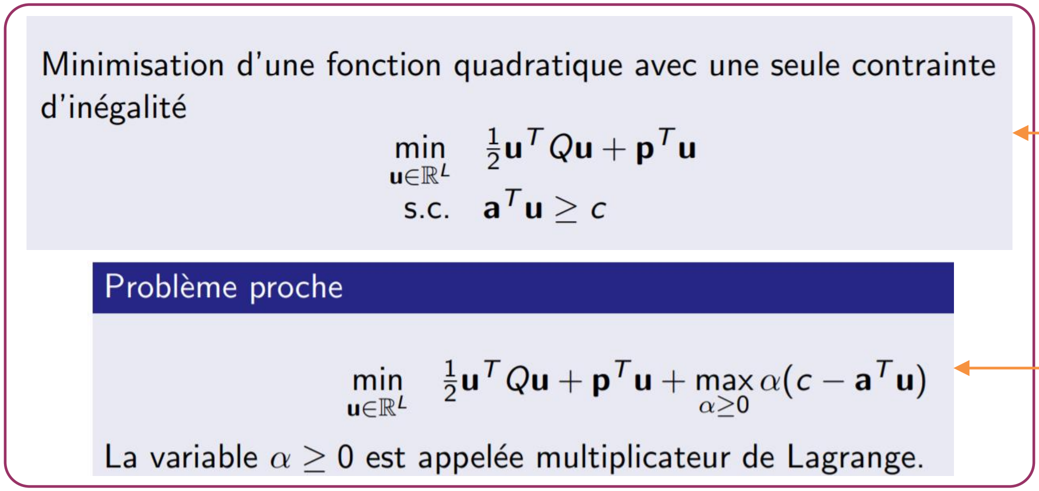
In order to take advantage of this trick, called **the kernel trick**, the SVM formulation must be reformulated so that the scalar product appears.

Non-linear SVM

Kernel trick - dual formulation

In order to take advantage of this trick, called **the kernel trick**, the SVM formulation must be reformulated so that the scalar product appears.

This can be done using **Lagrange multipliers**:



Minimisation d'une fonction quadratique avec une seule contrainte d'inégalité

$$\min_{\mathbf{u}\in\mathbb{R}^L} \quad \tfrac{1}{2}\mathbf{u}^T Q\mathbf{u} + \mathbf{p}^T\mathbf{u}$$
$$\text{s.c.} \quad \mathbf{a}^T\mathbf{u} \geq c$$

Problème proche

$$\min_{\mathbf{u}\in\mathbb{R}^L} \quad \tfrac{1}{2}\mathbf{u}^T Q\mathbf{u} + \mathbf{p}^T\mathbf{u} + \max_{\alpha\geq 0} \alpha(c - \mathbf{a}^T\mathbf{u})$$

La variable $\alpha \geq 0$ est appelée multiplicateur de Lagrange.

Quadratic minimisation with linear constraints

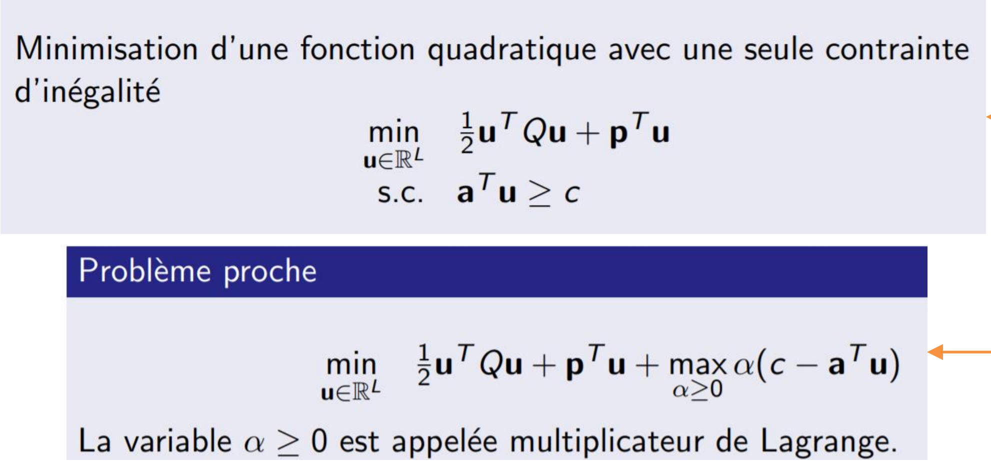
Equivalent representation

Non-linear SVM

Kernel trick - dual formulation

In order to take advantage of this trick, called **the kernel trick**, the SVM formulation must be reformulated so that the scalar product appears.

This can be done using **Lagrange multipliers**:



Minimisation d'une fonction quadratique avec une seule contrainte d'inégalité

$$\min_{\mathbf{u}\in\mathbb{R}^L} \quad \tfrac{1}{2}\mathbf{u}^T Q\mathbf{u} + \mathbf{p}^T\mathbf{u}$$
$$\text{s.c.} \quad \mathbf{a}^T\mathbf{u} \geq c$$

Problème proche

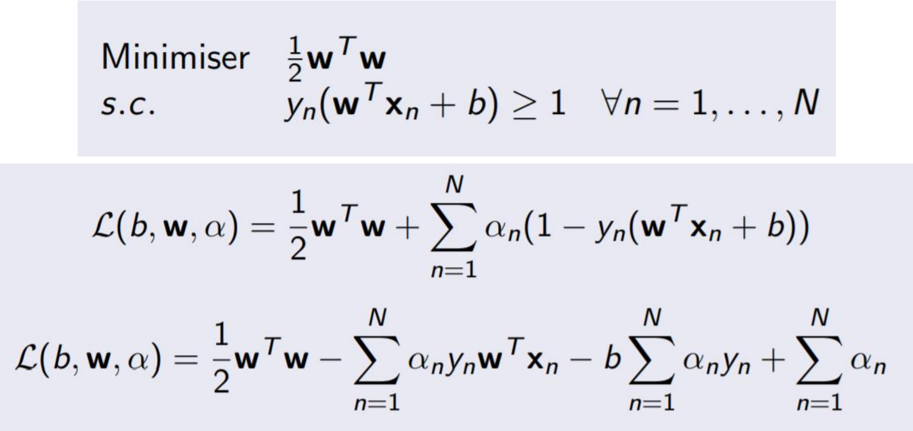
$$\min_{\mathbf{u}\in\mathbb{R}^L} \quad \tfrac{1}{2}\mathbf{u}^T Q\mathbf{u} + \mathbf{p}^T\mathbf{u} + \max_{\alpha\geq 0} \alpha(c - \mathbf{a}^T\mathbf{u})$$

La variable $\alpha \geq 0$ est appelée multiplicateur de Lagrange.

Quadratic minimisation with linear constraints

Equivalent representation

In the case of **SVM**:



$$\text{Minimiser} \quad \tfrac{1}{2}\mathbf{w}^T\mathbf{w}$$
$$\text{s.c.} \quad y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 \quad \forall n = 1, \dots, N$$

$$\mathcal{L}(b, \mathbf{w}, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{n=1}^{N} \alpha_n(1 - y_n(\mathbf{w}^T\mathbf{x}_n + b))$$

$$\mathcal{L}(b, \mathbf{w}, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{n=1}^{N} \alpha_n y_n \mathbf{w}^T\mathbf{x}_n - b\sum_{n=1}^{N} \alpha_n y_n + \sum_{n=1}^{N} \alpha_n$$

Dual formulation

Non-linear SVM

Kernel trick - dual formulation

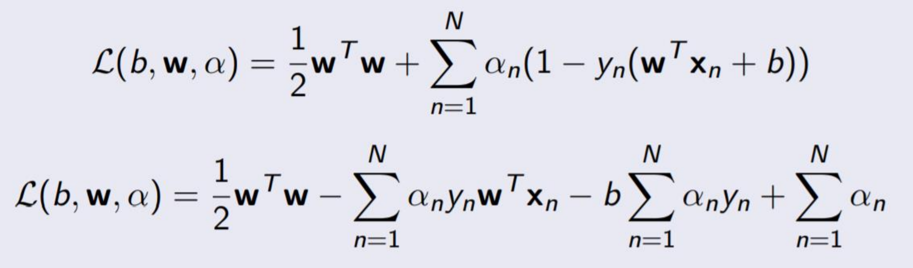
$$\mathcal{L}(b, \mathbf{w}, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{n=1}^{N} \alpha_n(1 - y_n(\mathbf{w}^T\mathbf{x}_n + b))$$

$$\mathcal{L}(b, \mathbf{w}, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{n=1}^{N} \alpha_n y_n \mathbf{w}^T\mathbf{x}_n - b\sum_{n=1}^{N} \alpha_n y_n + \sum_{n=1}^{N} \alpha_n$$

- We have transformed a **constrained** quadratic programming problem into... an **unconstrained** quadratic programming problem.
- This formulation has an important advantage in the case of SVMs (cf. Kernel trick)

Non-linear SVM

Kernel trick - dual formulation



$$\mathcal{L}(b,\mathbf{w},\alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{n=1}^{N}\alpha_n(1 - y_n(\mathbf{w}^T\mathbf{x}_n + b))$$

$$\mathcal{L}(b,\mathbf{w},\alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{n=1}^{N}\alpha_n y_n \mathbf{w}^T\mathbf{x}_n - b\sum_{n=1}^{N}\alpha_n y_n + \sum_{n=1}^{N}\alpha_n$$

- We have transformed a **constrained** quadratic programming problem into... an **unconstrained** quadratic programming problem.
- This formulation has an important advantage in the case of SVMs (cf. Kernel trick)

Interpretation :

- $\boldsymbol{\mathcal{L}}$ must be minimized with respect to the primal variables $(\boldsymbol{b}, \boldsymbol{w})$ and maximized with respect to the dual variables $\boldsymbol{\alpha_n}$.
- If the constraint is satisfied, i.e., $\mathbf{1} - \boldsymbol{y_n}\left(\mathbf{w}^{\mathbf{T}}\mathbf{x_n} + \mathbf{b}\right) \leq \mathbf{0}$ then $\boldsymbol{\alpha_n} = \mathbf{0}$, **except** for the support vectors (because increasing $\boldsymbol{\alpha_n}$ minimizes $\boldsymbol{\mathcal{L}}$ which is contrary to the objective).
- If the constraint is violated: $\boldsymbol{\alpha_n}$ will grow to maximize, and $\boldsymbol{\mathcal{L}}(b, w)$ will decrease so that the constraints are satisfied.

Non-linear SVM

Kernel trick – minimisation % (b,w)

$$\mathcal{L}(b,\mathbf{w},\alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{n=1}^{N}\alpha_n(1 - y_n(\mathbf{w}^T\mathbf{x}_n + b))$$

$$\mathcal{L}(b,\mathbf{w},\alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{n=1}^{N}\alpha_n y_n \mathbf{w}^T\mathbf{x}_n - b\sum_{n=1}^{N}\alpha_n y_n + \sum_{n=1}^{N}\alpha_n$$

minimisation % (b,w)

$$\frac{\partial\mathcal{L}}{\partial b} = 0 \quad \Rightarrow \quad \textstyle\sum_{n=1}^{N}\alpha_n y_n = 0$$

$$\frac{\partial\mathcal{L}}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \textstyle\sum_{n=1}^{N}\alpha_n y_n \mathbf{x}_n$$

We replace the values obtained in the Lagrangian and we obtain

$$\mathcal{L}(\alpha) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \mathbf{x}_n^T\mathbf{x}_m + \sum_{n=1}^{N}\alpha_n$$

A formulation to maximize that depends only on the dual variable α_n

Non-linear SVM

Kernel trick - dual formulation

The optimization problem can then be written as :

$$
\begin{array}{ll}
\min\limits_{\alpha\in\mathbb{R}^N} & \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \mathbf{x}_n^T \mathbf{x}_m - \sum_{n=1}^{N}\alpha_n \\
s.c. & \sum_{n=1}^{N} y_n \alpha_n = 0 \\
 & \alpha_n \geq 0
\end{array}
$$

Thus, we have the following decision function:

$$
g(\mathbf{x}) = sign\ \Big(\sum_{\alpha_n^*>0} y_n \alpha_n^* \mathbf{x}_n^T \mathbf{x} + b^*\Big)
$$

- Only the support vectors are needed to predict the class of a new data, as they define the separating hyperplane.
- The calculation of the $\boldsymbol{\alpha_n}$ is done using a solver.

Non-linear SVM

Kernel trick - dual formulation

The optimization problem can then be written as :

$$\begin{aligned} \min_{\alpha \in \mathbb{R}^N} \quad & \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \Phi(\mathbf{x}_n)^T \Phi(\mathbf{x}_m) - \sum_{n=1}^{N} \alpha_n \\ s.c. \quad & \sum_{n=1}^{N} y_n \alpha_n = 0 \\ & \alpha_n \geq 0 \end{aligned}$$

Thus, we have the following decision function:

$$g(\mathbf{x}) = sign\ \left(\sum_{\alpha_n^* > 0} y_n \alpha_n^* \Phi(\mathbf{x}_n)^T \Phi(\mathbf{x}) + b^* \right)$$

- Only the support vectors are needed to predict the class of a new data, as they define the separating hyperplane.
- The calculation of the $\boldsymbol{\alpha_n}$ is done using a solver.
- This new formulation allows to benefit from the kernel trick.

Non-linear SVM

Kernel trick - dual formulation

In the previous formulation (for learning and decision), the only calculation that is performed in the redescription space is a scalar product.

$$\boldsymbol{\phi}(\boldsymbol{x}).\boldsymbol{\phi}(\boldsymbol{x}')$$

The idea is to find a function that computes both the transformation and the scalar product.

$$\boldsymbol{K}_{\boldsymbol{\phi}}(\boldsymbol{x}.\boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x}).\boldsymbol{\phi}(\boldsymbol{x}')$$

$\boldsymbol{K}_{\boldsymbol{\phi}}$ is a kernel function.

Non-linear SVM

Kernel trick - dual formulation

In the previous formulation (for learning and decision), the only calculation that is performed in the redescription space is a scalar product.

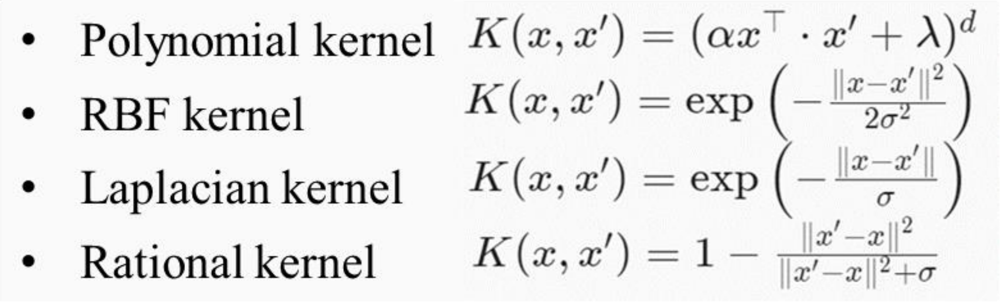
$$\boldsymbol{\phi}(\boldsymbol{x}).\boldsymbol{\phi}(\boldsymbol{x}')$$

The idea is to find a function that computes both the transformation and the scalar product.

$$\boldsymbol{K}_{\boldsymbol{\phi}}(\boldsymbol{x}.\boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x}).\boldsymbol{\phi}(\boldsymbol{x}')$$

$\boldsymbol{K}_{\boldsymbol{\phi}}$ is a kernel function.

A number of kernel functions are known, including :

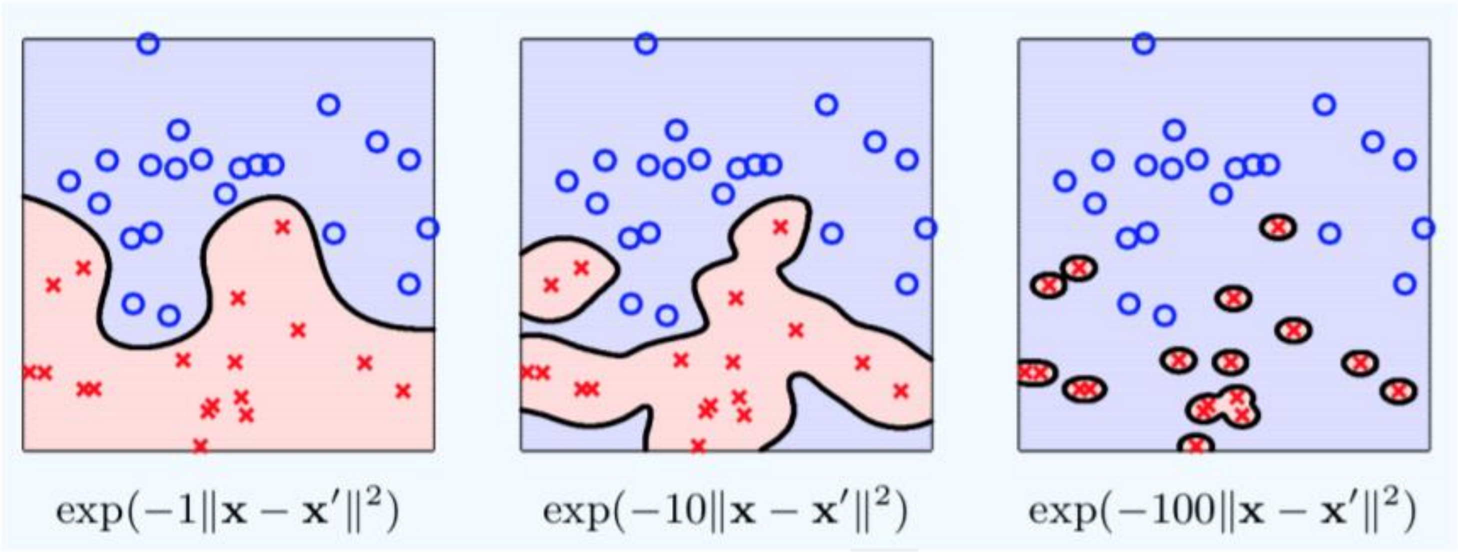


- Polynomial kernel $K(x, x') = (\alpha x^\top \cdot x' + \lambda)^d$
- RBF kernel $K(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$
- Laplacian kernel $K(x, x') = \exp\left(-\frac{\|x-x'\|}{\sigma}\right)$
- Rational kernel $K(x, x') = 1 - \frac{\|x'-x\|^2}{\|x'-x\|^2+\sigma}$

Non-linear SVM

Kernel trick - Gaussian Kernel (RBF)

Illustration of classification by RBF kernel



$\exp(-1\|\mathbf{x}-\mathbf{x}'\|^2)$

$\exp(-10\|\mathbf{x}-\mathbf{x}'\|^2)$

$\exp(-100\|\mathbf{x}-\mathbf{x}'\|^2)$

Conclusion

- Does not allow the extraction of a comprehensible/interpretable model.
- Unsuitable for mining very large volumes of data.
- Kernel selection: no solution at the moment except by trial and error.
- Provides a global optimum, but what does it really mean for a real dataset?
- To do machine learning, you need to understand the algorithm you are using. Clearly, a good understanding of SVM is not easy.

Summary

For a first use of SVM, you have to keep in mind that :

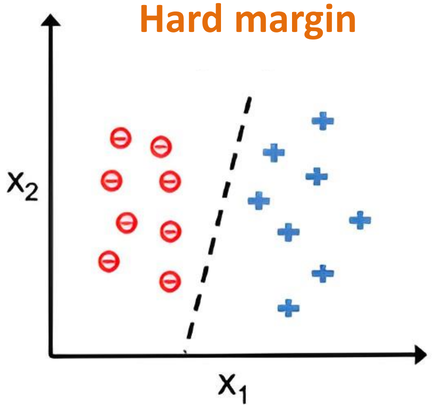
SVM is used for binary classification problems

Summary

For a first use of SVM, you have to keep in mind that :

SVM is used for binary classification problems

Hard margin



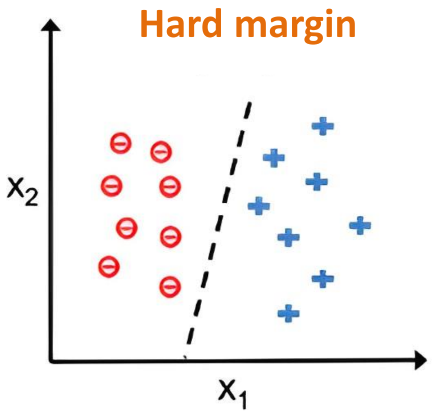
Only works when the data can be separated linearly.

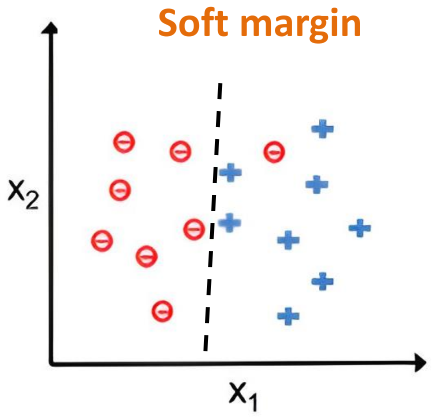
No hyperparameters defined in the formulation.

Summary

For a first use of SVM, you have to keep in mind that :

SVM is used for binary classification problems





Only works when the data can be separated linearly.

No hyperparameters defined in the formulation.

Allows to tolerate classification errors during the training phase.

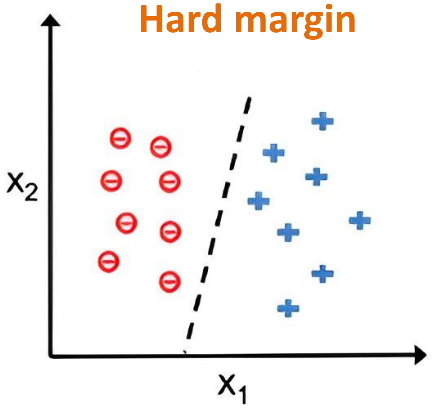
The **hyperparameter C** for controlling the margin width, has to be optimized.

Summary

For a first use of SVM, you have to keep in mind that :

SVM is used for binary classification problems

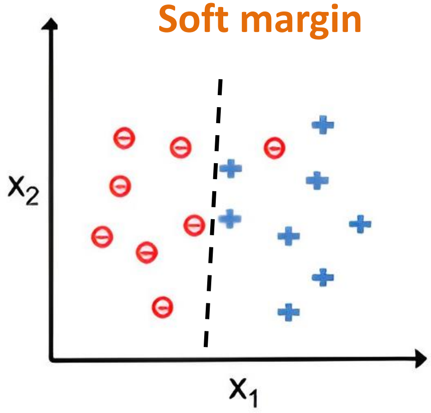
Hard margin



Only works when the data can be separated linearly.

No hyperparameters defined in the formulation.

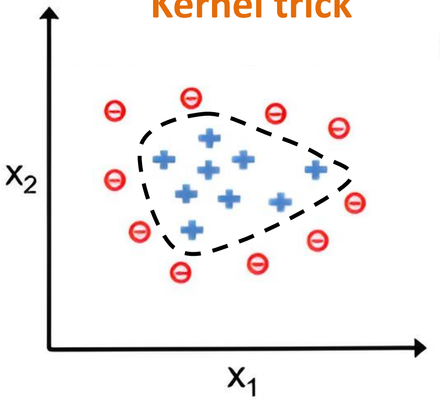
Soft margin



Allows to tolerate classification errors during the training phase.

The **hyperparameter C** for controlling the margin width, has to be optimized.

Kernel trick



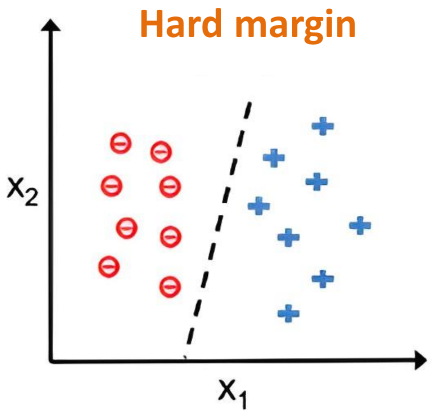
Allows to create non-linear separations.

The **Kernel function** and **its parameters** are the hyperparameters to be optimized.

Summary

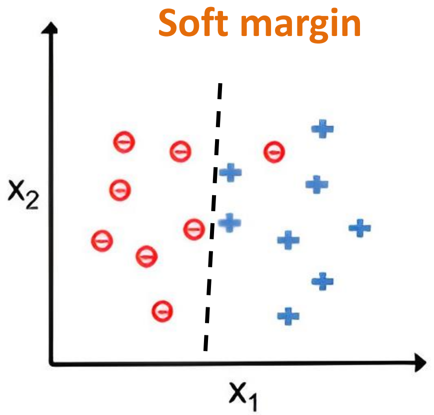
For a first use of SVM, you have to keep in mind that :

SVM is used for binary classification problems


Hard margin

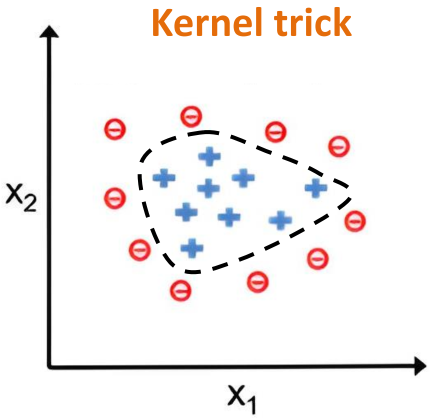
Only works when the data can be separated linearly.

No hyperparameters defined in the formulation.


Soft margin

Allows to tolerate classification errors during the training phase.

The **hyperparameter C** for controlling the margin width, has to be optimized.


Kernel trick

Allows to create non-linear separations.

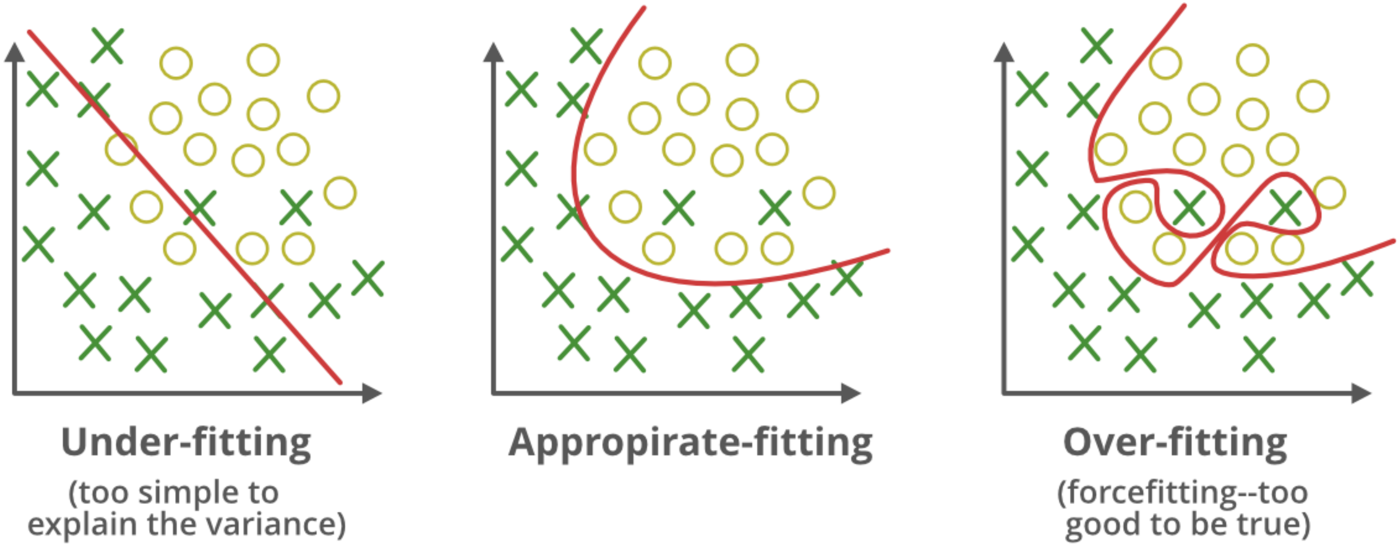
The **Kernel function** and **its parameters** are the hyperparameters to be optimized.

In general, both the **C-hypeparameter** and the **kernel function** are incorporated in the same SVM formulation.

SVM performance optimisation

Hyperparameters

The aim of **hyperparameters** is to control the over-fitting and under-fitting of the model.


Under-fitting
(too simple to explain the variance)
Appropirate-fitting
Over-fitting
(forcefitting--too good to be true)

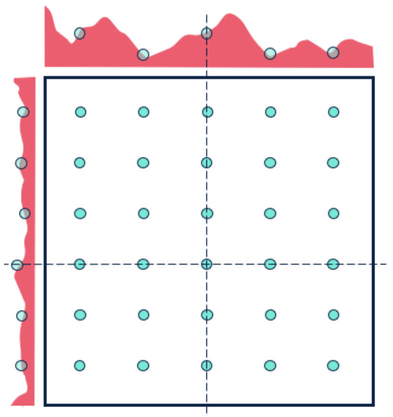
Optimal hyperparameters often differ for different datasets.

To get the best hyperparameters the following steps are followed:

1. For each proposed hyperparameter setting the model is evaluated.
2. The hyperparameters that give the best model are selected.

SVM performance optimisation

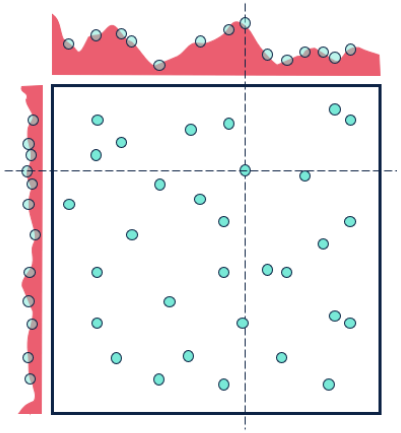
Hyperparameters Search



Grid search

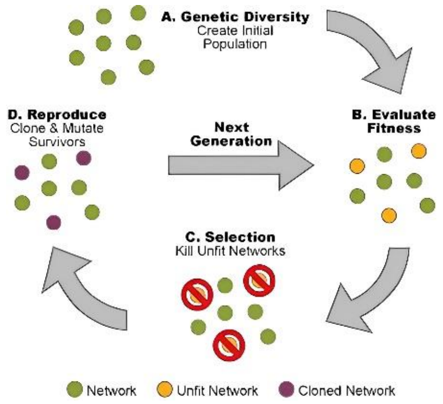
Grid search involves specifying a list of possible hyperparameter values that you want to test.

The algorithm will then train models with all possible combinations of the provided hyperparameter values and evaluate the performance of each trained model using a specified metric.



Random search

Instead of providing a list of hyperparameter values, you will provide statistical distributions of hyperparameter values from which you want the optimization algorithm to test values.



Genetic algorithm

One of the most widely used metaheuristic algorithms is the genetic algorithm (GA), which is based on the evolutionary idea that individuals (hyperparameters in this case) with the highest potential for survival and adaptation to the environment are more likely to survive and pass on their qualities to future generations.

SVM performance optimisation

Genetic Algorithm (GA)

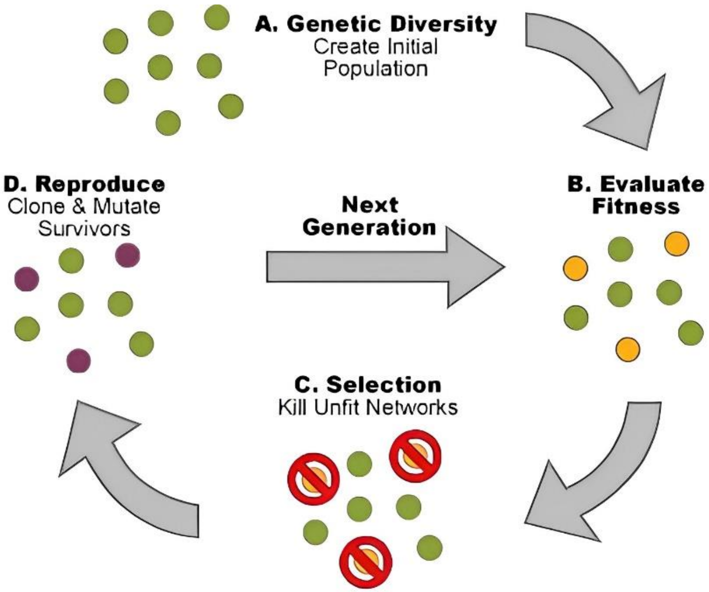
The algorithm starts by **randomly generating** many different initial sets of hyperparameter values.

The algorithm then trains models with these randomly generated hyperparameter values and **assesses** them with a defined **evaluation metric**.

The sets of hyperparameter values that perform the worst on the evaluation metric are **thrown out**, and **new values are generated** to replace them based on the values that performed best on the evaluation metric.

In this way genetic algorithms try to mimic the human evolution to some extent.

However, to successfully implement GA, its basic structure and steps must be clearly defined.

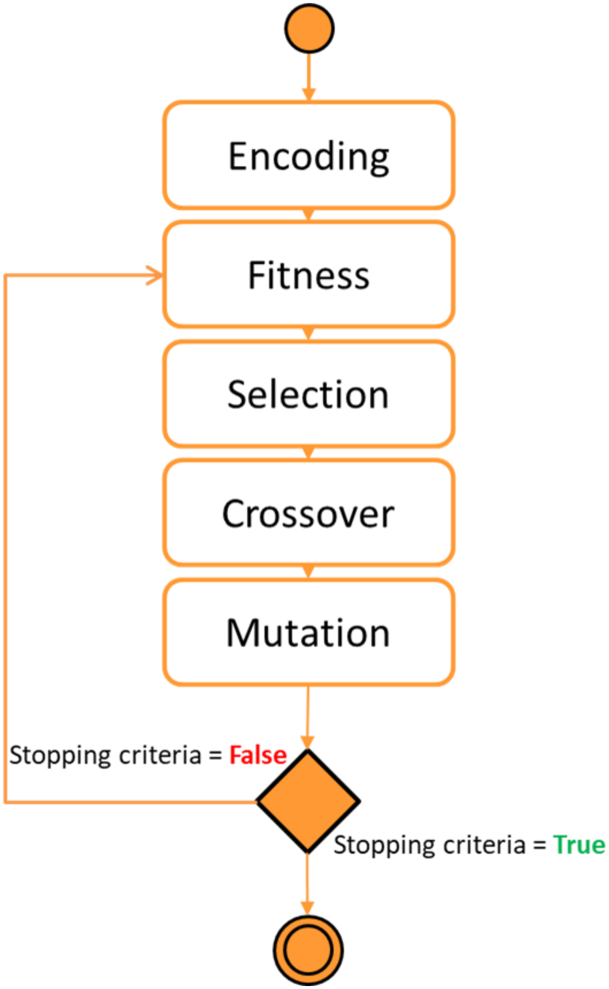


SVM performance optimisation

Genetic Algorithm (GA)

The algorithm works by first creating a random population of a fixed size.

The main loop of the algorithm is repeated for a **fixed number** of iterations or until a termination criteria is reached.



Encoding and population initialization : One of the most critical decisions that significantly affects the performance of a genetic algorithm while implementing it, is deciding the representation to use in order to represent the candidate solutions (individuals).

Two well-known representations are :

- Binary representation

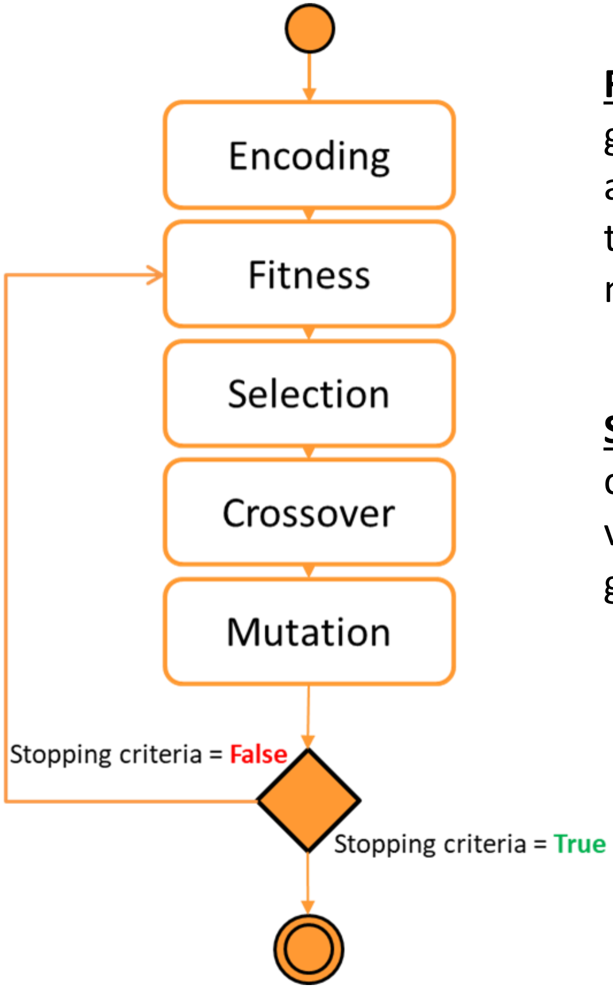
0	0	1	0	1	1	1	0	0	1

- Integer representation

1	2	3	4	3	2	4	1	2	1

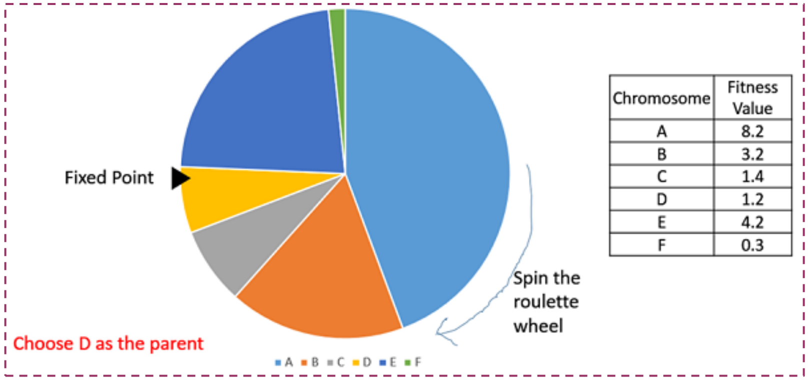
SVM performance optimisation

Genetic Algorithm (GA)



Fitness : Through any GA, it is necessary to be able to evaluate how good a potential solution is, compared to other solutions. To do so, a fitness function is defined. The fitness values are then used for the selection of individuals (parents), on which crossover and mutation operations will be applied.

Selection : Selection operation in a genetic algorithm consists of choosing individuals (parents) for reproduction. Parent selection is very important to the convergence of the genetic algorithm (GA), as good parents generate and yield better and fitter solutions.

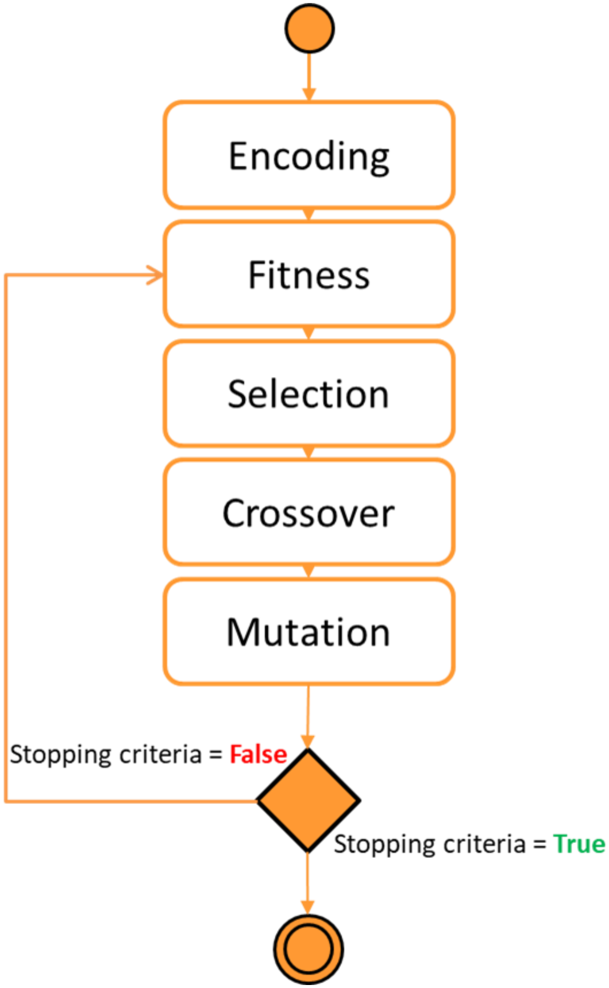


Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Roulette Wheel Selection

SVM performance optimisation

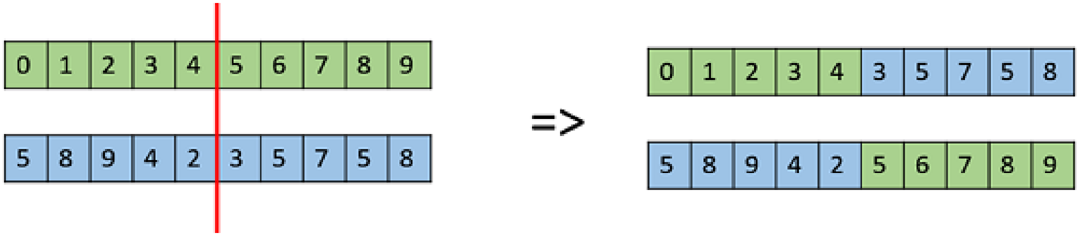
Genetic Algorithm (GA)



Crossover : By this operator, individuals (parents) share information by crossing their genotypes to create better individuals (children).

- *One Point Crossover*

In this one-point crossover, a random crossover point is selected and the tails of its two parents are swapped to get new off-springs.



0 1 2 3 4 | 5 6 7 8 9

5 8 9 4 2 | 3 5 7 5 8

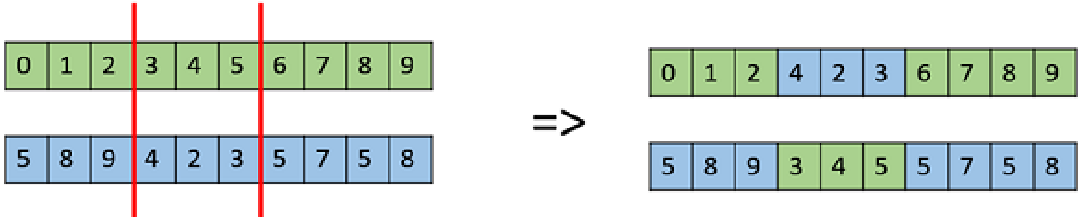
=>

0 1 2 3 4 3 5 7 5 8

5 8 9 4 2 5 6 7 8 9

- *Multi Point Crossover*

Multi point crossover is a generalization of the one-point crossover wherein alternating segments are swapped to get new off-springs.



0 1 2 | 3 4 5 | 6 7 8 9

5 8 9 | 4 2 3 | 5 7 5 8

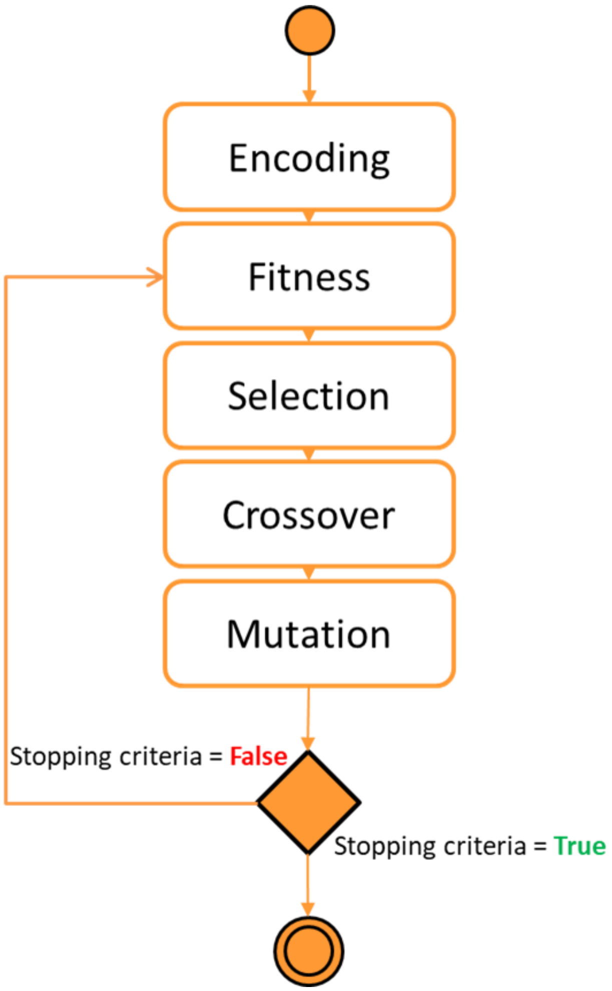
=>

0 1 2 4 2 3 6 7 8 9

5 8 9 3 4 5 5 7 5 8

SVM performance optimisation

Genetic Algorithm (GA)



Mutation : Mutation may be defined as a small random modification in the chromosome to get a new solution. It is used to introduce diversity in the genetic population and to release from local minima.

- *Bit Flip Mutation*

In this bit flip mutation, we select one or more random bits and flip them. This is used for binary encoded GAs.

0	0	1	1	0	1	0	0	1	0

=>

0	0	1	0	0	1	0	0	1	0

- *Swap Mutation*

In swap mutation, we select two positions on the chromosome at random, and interchange the values. This is common in permutation based encodings.

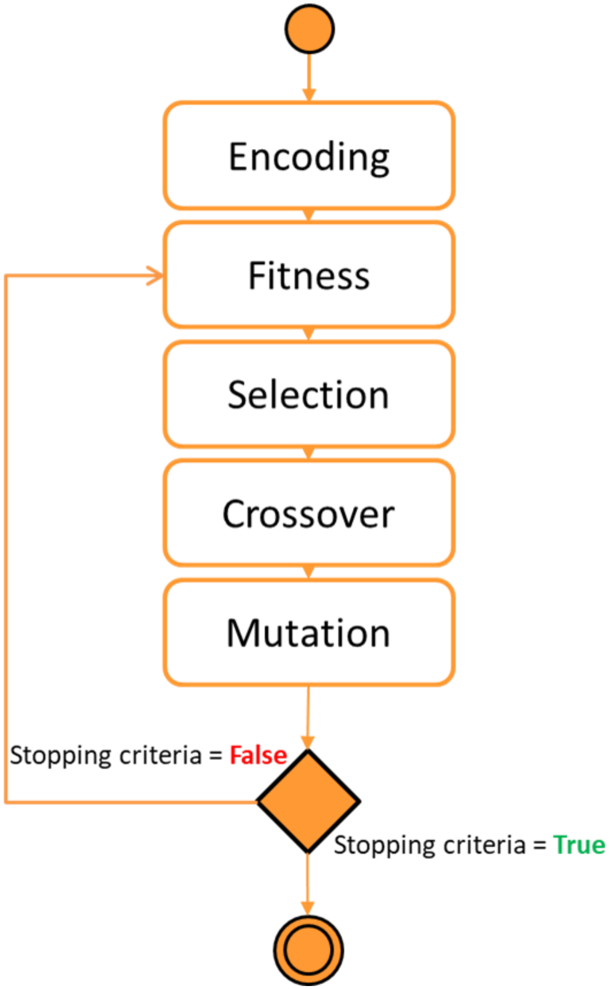
1	2	3	4	5	6	7	8	9	0

=>

1	6	3	4	5	2	7	8	9	0

SVM performance optimisation

Genetic Algorithm (GA)



Stopping criteria : GA progresses very quickly with better solutions over the course of a few iterations, but this tends to saturate in the later stages where the improvements are very small.

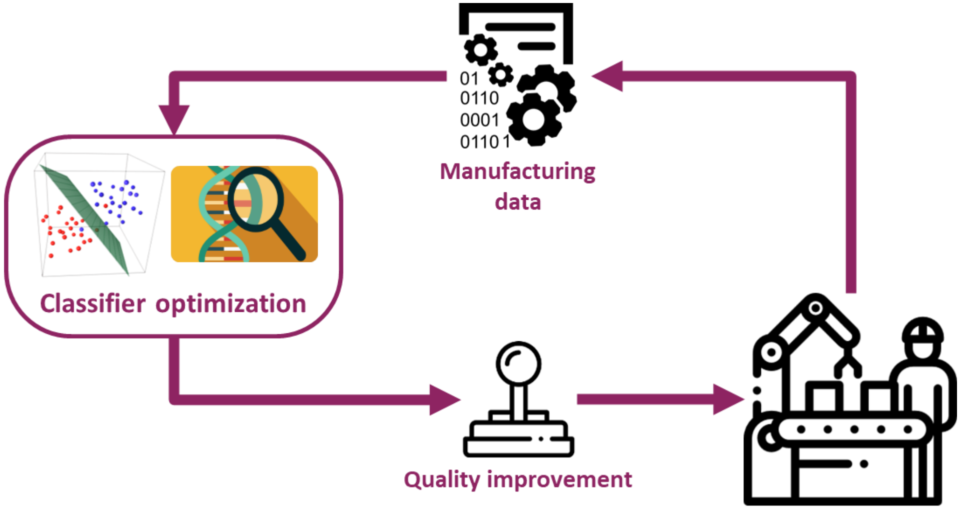
Typically, we keep one of the following termination conditions.

- When there has been no improvement in the population for X iterations.
- When we reach an absolute number of generations.
- When the value of the objective function has reached a certain predefined value.

Project

Project – issue 1

How to **optimize** the predictive performance of **SVM**, in order to improve the **manufacturing systems quality** ?



In this project, **you won't be focusing on** :

- × The implementation of SVM : *a python code is already available on SAVOIR.*
- × The choice of the dataset : *provided on SAVOIR.*

On the other hand, **you are going to focus on** :

- ✓ The implementation of GA for the optimization of SVM with an RBF Kernel.
- ✓ The application of the developed GA algorithm on manufacturing data.
- ✓ The assessment and the analysis of the performance of the developed GA algorithm.

A step-by-step guide for the implementation of GA is provided in the following slides.

Project steps

Encoding

SVM hyperparameters to encode

- Regularization parameter C
- RBF kernel *(contains only one hyperparameter)*

Kernel	Expression	Parameters to tune	Variation range
RBF	$e^{\left(-\frac{\|\|x-y\|\|^2}{2\sigma^2}\right)}$	Penalty C Parameter σ	[1, 1000] [0.001, 10]

Genetic representation : integer representation

RBF kernel	A0	A1	A2	A3	A4	A5	A6
Variation ranges	[0-9]	[0-9]	[0-9]	[0-9]	[0-9]	[0-9]	[0-9]
	↓	↓	↓	↓	↓	↓	↓
Mapping	A0*100	+ A1*10	+ A2 + 1	A3 +	A4*0,1 +	A5*0,01 +	(A6+1)*0,001
Solution	C			σ			

Project steps

Encoding

Genetic representation : integer representation

RBF kernel	A0	A1	A2	A3	A4	A5	A6
Variation ranges	[0-9]	[0-9]	[0-9]	[0-9]	[0-9]	[0-9]	[0-9]
	↓	↓	↓	↓	↓	↓	↓
Mapping	A0*100	+ A1*10	+ A2 + 1	A3 + A4*0,1 + A5*0,01 + (A6+1)*0,001			
Solution	C			σ			

Population initialization : random generation

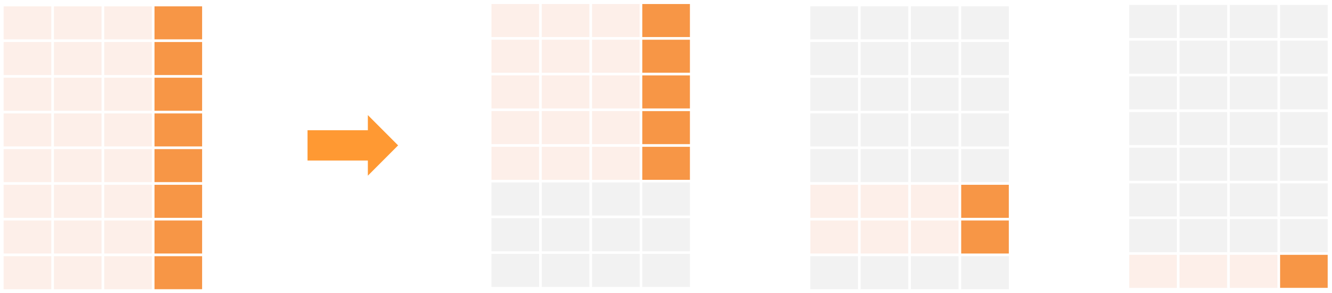
8	4	1	2	9	4	8
6	7	9	4	6	1	9
3	2	6	7	2	0	1
			⋮			
0	4	7	5	4	3	6

Project steps

Fitness evaluation

Fitness

1. Split the dataset into training, validation and test sets.



Dataset

Training

Validation

Test

2. Train a SVM model using the encoded hyperparameters of the chromosome.

8	4	1	2	9	4	8
↓	↓	↓	↓	↓	↓	↓

8*100 + 4*10 + 1 + 1

2 + 9*0,1 + 4*0,01 + (8+1)*0,001

842	**2,949**

3. Evaluate the fitness of the chromosome.

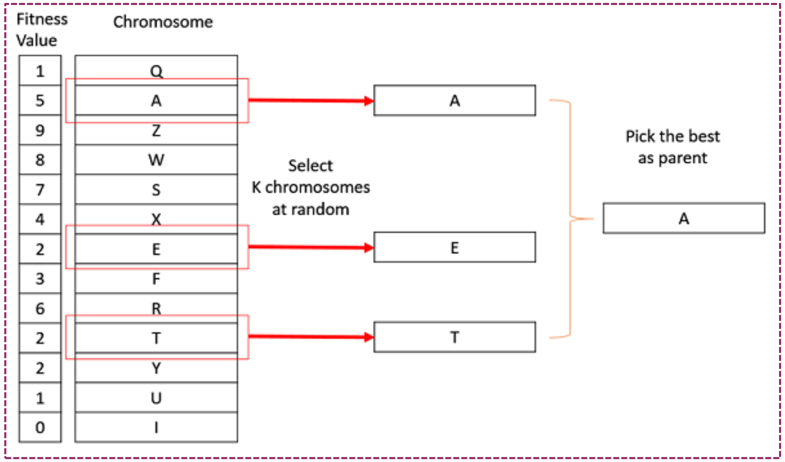
Fitness = prediction accuracy of validation set

Project steps

Selection

Parents selection : tournament selection

Selection operation consists of choosing individuals (parents) for reproduction

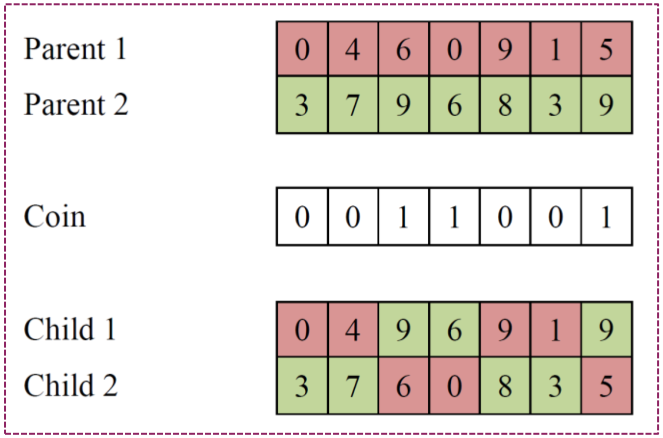


Fitness Value	Chromosome
1	Q
5	A
9	Z
8	W
7	S
4	X
2	E
3	F
6	R
2	T
2	Y
1	U
0	I

Crossover

Crossover : uniform crossover

Consist of flipping a coin for each gene to decide whether or not it'll be included in the off-spring.

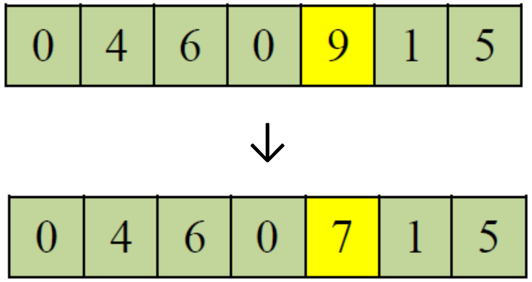


Parent 1	0	4	6	0	9	1	5
Parent 2	3	7	9	6	8	3	9
Coin	0	0	1	1	0	0	1
Child 1	0	4	9	6	9	1	9
Child 2	3	7	6	0	8	3	5

Project steps

Mutation: random resetting

A random value from the set of permissible values is assigned to a randomly chosen gene.



0	4	6	0	9	1	5

↓

0	4	6	0	7	1	5

Mutation

Stopping criteria

The algorithm stops running after a predefined number of iterations.

Stopping criteria

Plan

Learning outcomes:

- **How to evaluate and handle the impact of noise (uncertainties) in manufacturing data on the performance of classification techniques [SVM as a case study]?**

Keywords : *Support Vector Machine, Measurment uncertainties, Genetic algorithm, Monte-Carlo simulation, Sobol sensitivity analysis, Robust optimisation.*

1st issue :

- **Objective :** How to **optimise** the prediction accuracy of SVM ?

  ⟶ Introduction to SVM.

  ⟶ A genetic algorithm for SVM prediction accuracy optimisation.

2nd issue :

- **Objective :** How to **evaluate** the impact of uncertainties on SVM accuracy ?

  ⟶ Monte-Carlo simulation for uncertainties impact quantification.

  ⟶ Sobol sensitivity analysis for key uncertainties identification.

3rd issue :

- **Objective :** How to **mitigate** the impact of uncertainties on SVM accuracy ?

  ⟶ Bibliographical study.

Robustness study

Robustness ?

The ability of a system to maintain its performance, despite changes in operating conditions or the **presence of uncertainties** related to its parameters or components.

A product or a process whose performance is **little affected by the variation** of the conditions in which it operates is said to be **robust**.

Robustness is, according to **G. Taguchi**, the essential component of the quality of a product.

The principle of robust engineering is to tackle the variability experienced in production, not by eliminating the causes, which is generally an impossible or too costly solution, but by seeking a design/modeling that limits its effects.

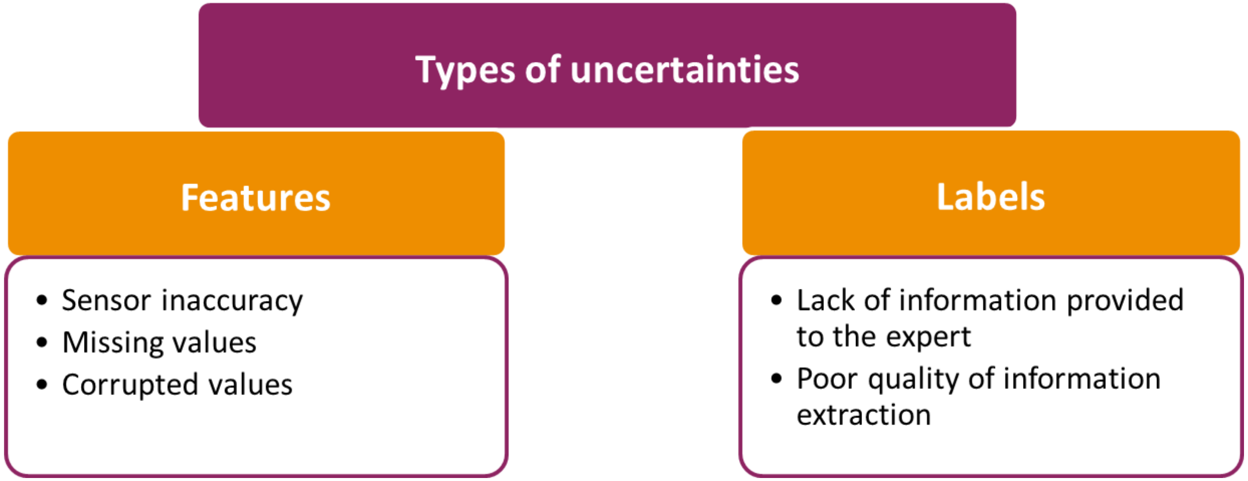
Robustness study

Robustness of an ML model

Robustness is also desired in Machine Learning, where we seek to implement models that are robust to uncertainties.

Measurable data for machine learning and data mining are usually imprecise, incomplete, or noisy.

Machine learning systems must deal with imperfect data and are therefore expected to reason under conditions of ignorance.



Types of uncertainties

Features

- Sensor inaccuracy
- Missing values
- Corrupted values

Labels

- Lack of information provided to the expert
- Poor quality of information extraction

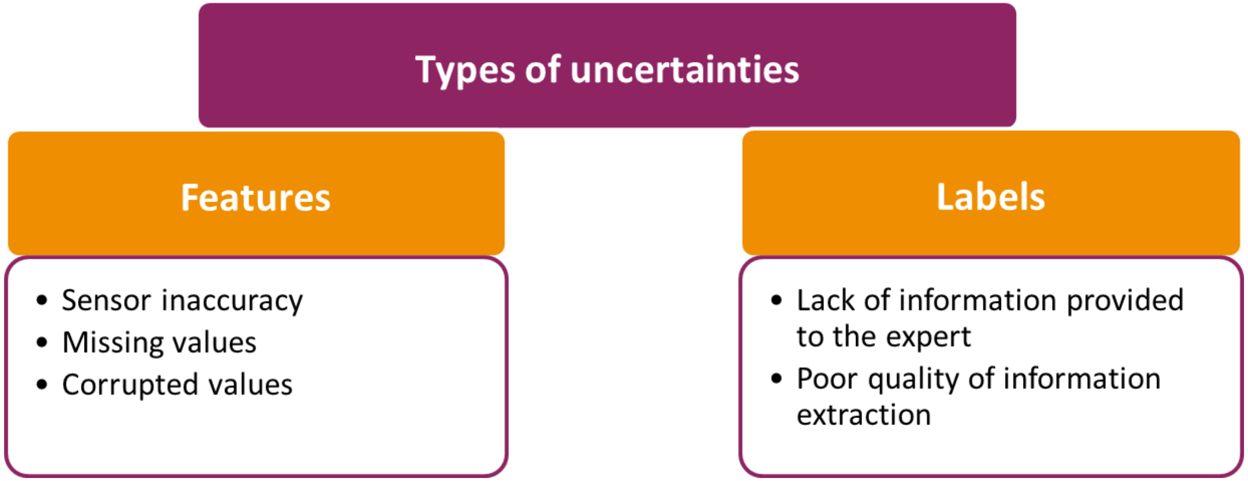
Robustness study

Robustness of an ML model

Robustness is also desired in Machine Learning, where we seek to implement models that are robust to uncertainties.

Measurable data for machine learning and data mining are usually imprecise, incomplete, or noisy.

Machine learning systems must deal with imperfect data and are therefore expected to reason under conditions of ignorance.



Types of uncertainties

Features

- Sensor inaccuracy
- Missing values
- Corrupted values

Labels

- Lack of information provided to the expert
- Poor quality of information extraction

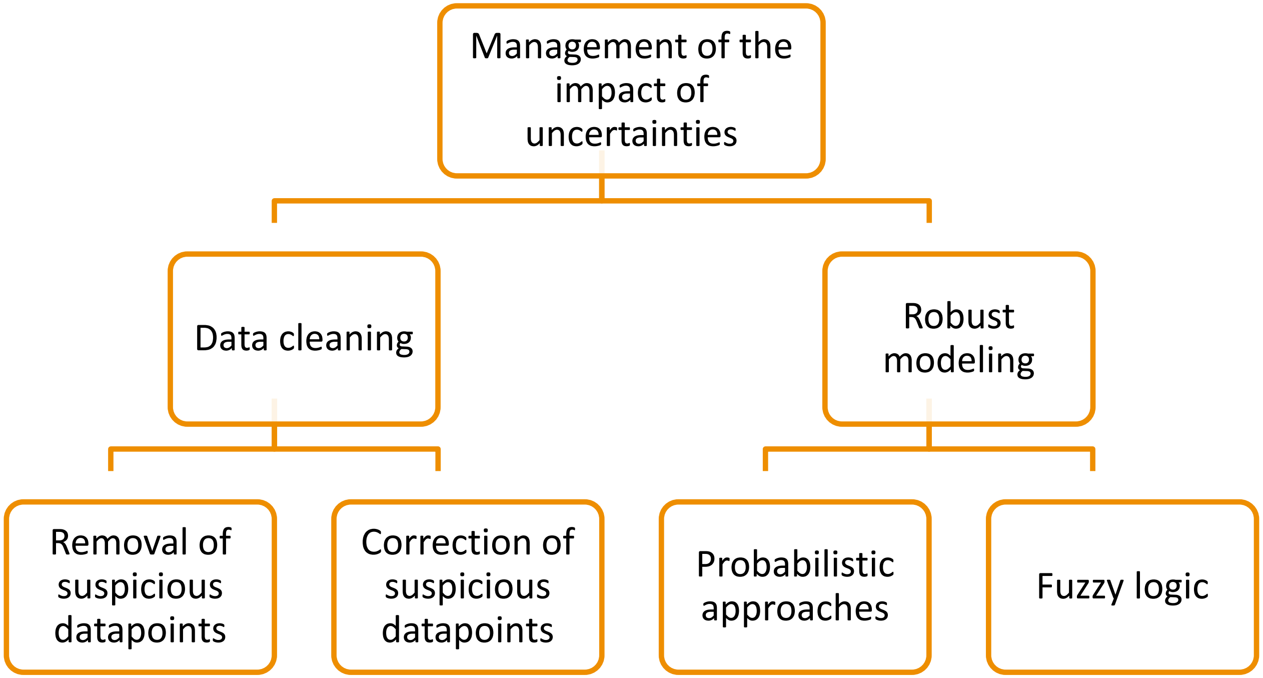
Impact of uncertainties:

- Decreases the performance of the classifiers.
- Increases the amount of data needed for training.

Robustness study

Robustness of an ML model

To manage the impact of uncertainties on machine learning techniques.



Management of the impact of uncertainties

Data cleaning

Robust modeling

Removal of suspicious datapoints

Correction of suspicious datapoints

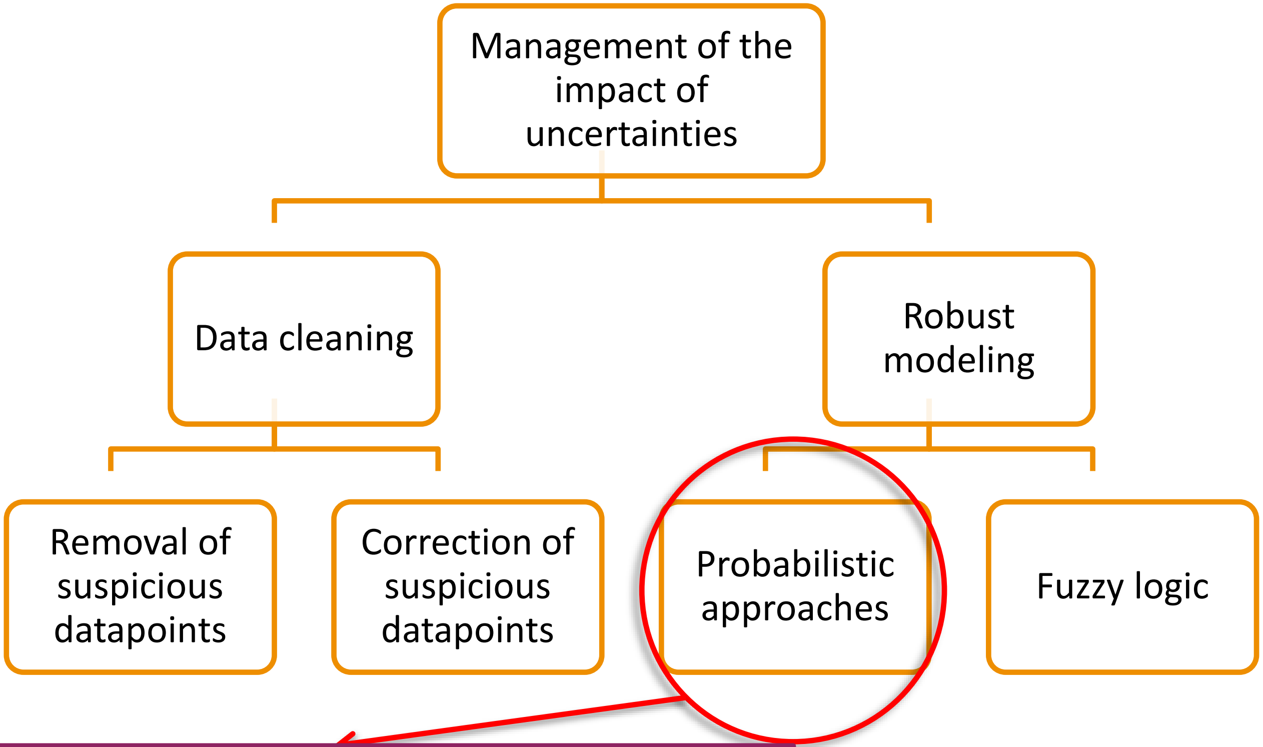
Probabilistic approaches

Fuzzy logic

Robustness study

Robustness of an ML model

To manage the impact of uncertainties on machine learning techniques.



Management of the impact of uncertainties
Data cleaning
Robust modeling
Removal of suspicious datapoints
Correction of suspicious datapoints
Probabilistic approaches
Fuzzy logic

$$
\begin{array}{lll}
\min & \frac{1}{2}w^T.w + C\sum_{i=1}^{n}\xi_i & \\
\text{s.c} & y_i(w^T.x_i + b) \geq 1 - \xi_i & \forall\, i = 1, \ldots \ldots, n \\
 & \xi_i \ \geq 0 & \forall\, i = 1, \ldots \ldots, n
\end{array}
$$

$$\Downarrow$$

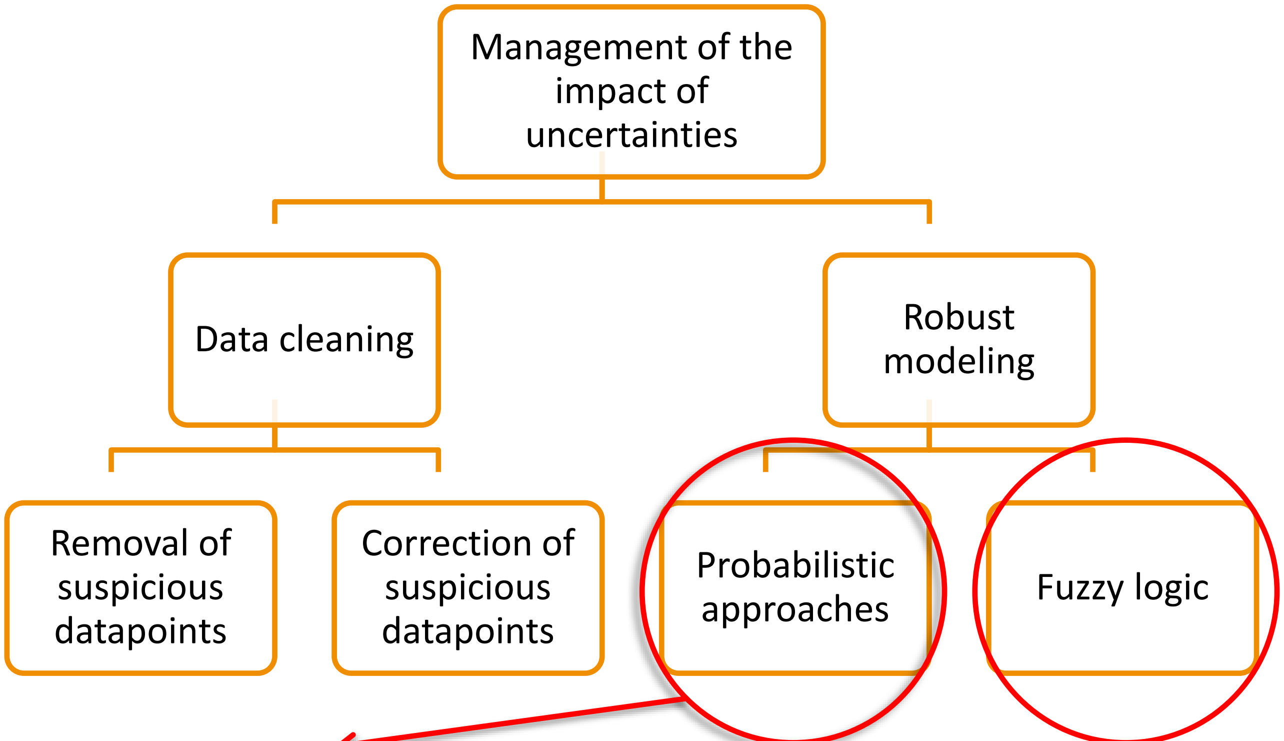
$$
\begin{array}{lll}
\min & \frac{1}{2}w^T.w + C\sum_{i=1}^{n}\xi_i & \\
\text{s.c} & y_i(w^T.(x_i + \Delta x_i) + b) \geq 1 - \xi_i & \forall\, i = 1, \ldots \ldots, n \\
 & \xi_i \ \geq 0 & \forall\, i = 1, \ldots \ldots, n \\
 & ||\Delta x_i|| \leq \delta_i & \forall\, i = 1, \ldots \ldots, n
\end{array}
$$

[Bi & Zhang 2004]

Robustness study

Robustness of an ML model

To manage the impact of uncertainties on machine learning techniques.



Management of the impact of uncertainties

Data cleaning

Robust modeling

Removal of suspicious datapoints

Correction of suspicious datapoints

Probabilistic approaches

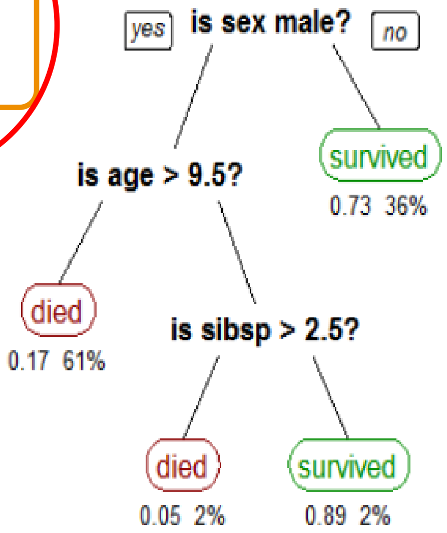
Fuzzy logic

$$\begin{array}{lll} \min & \frac{1}{2} w^T.w + C \sum_{i=1}^{n} \xi_i & \\ \text{s.c} & y_i(w^T.x_i + b) \geq 1 - \xi_i & \forall\, i = 1, \dots \dots, n \\ & \xi_i \ \geq 0 & \forall\, i = 1, \dots \dots, n \end{array}$$

$$\downarrow$$

$$\begin{array}{lll} \min & \frac{1}{2} w^T.w + C \sum_{i=1}^{n} \xi_i & \\ \text{s.c} & y_i(w^T.(x_i + \Delta x_i) + b) \geq 1 - \xi_i & \forall\, i = 1, \dots \dots, n \\ & \xi_i \ \geq 0 & \forall\, i = 1, \dots \dots, n \\ & ||\Delta x_i|| \leq \delta_i & \forall\, i = 1, \dots \dots, n \end{array}$$

[Bi & Zhang 2004]



[V. A. upta 2014]

Robustness study

Robustness of an ML model

A first objective is not to control the impact of uncertainties on the performance of machine learning techniques, but to :

1. **Quantify the impact of these uncertainties on the performance of classifiers**.
2. **Identify the parameters that contribute most to this impact.**

Robustness study

Robustness of an ML model

A first objective is not to control the impact of uncertainties on the performance of machine learning techniques, but to :

1. **Quantify the impact of these uncertainties on the performance of classifiers**.

How to ?

Monte Carlo simulation to calculate the impact of measurement uncertainties on **prediction accuracy**

Quantifying the impact of uncertainties

Monte Carlo simulation

The term Monte Carlo refers to a family of algorithmic methods that use random processes to calculate an approximate numerical value.

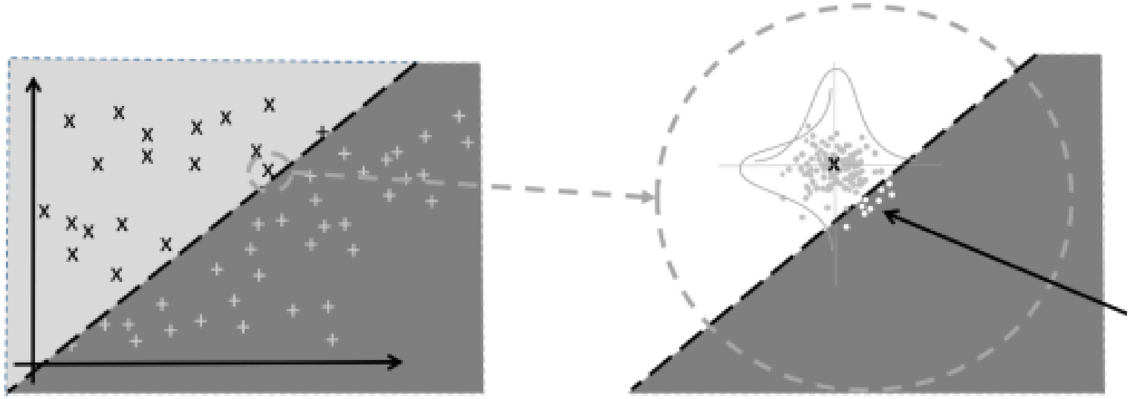
The main purpose of Monte Carlo simulation is to perform a large number of experiments on random samples and then draw conclusions about the model outputs.

Monte Carlo simulations are considered effective mathematical analysis methods used to solve complex engineering problems because of their ability to handle large numbers of random variables, different types of distributions, and highly nonlinear engineering models.

Quantifying the impact of uncertainties

Monte Carlo simulation

Illustration of a data point subject to measurement uncertainties :



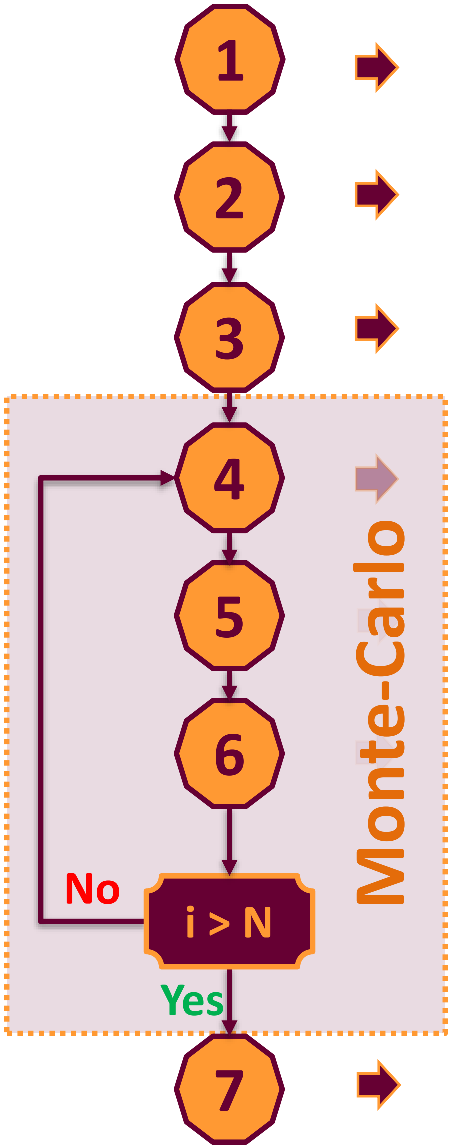
The approach to evaluate the impact of uncertainties on the predictive performance of a classifier consists in:

1. Set up a predictive model.
2. Compare the prediction of the model on a **clean** validation/test set to its prediction on **noisy** validation/test sets.

Quantifying the impact of uncertainties

Monte-Carlo for impact quantification

A procedure for evaluating the robustness of classifiers; **SVM** as a case study.



1. Preprocess the dataset : data split, data normalization, …
2. Train the SVM model using the training set.
3. Generate 1000 (**N**) random Gaussian noises.
4. Noise the validation/test set with the i^{th} noise.
5. Assess the prediction accuracy of SVM.
6. Store the found result.
7. Quantify the impact of uncertainties.

Quantifying the impact of uncertainties

Example

Application to an industrial dataset

SVM hyperparameters :

- Kernel function
- Kernel hyperparamters.
- The **C** hyperparameter.

Parameter of Monte-Carlo simulation:

- Noise magnitude : 2,5%
- Sample size : 1000

Quantifying the impact of uncertainties

Example

Application to an industrial dataset

SVM hyperparameters :

- Kernel function
- Kernel hyperparamters.
- The **C** hyperparameter.

Parameter of Monte-Carlo simulation:

- Noise magnitude : 2,5%
- Sample size : 1000

Dataset	Number of features	Size of class 1	Size of class 2	Manufacturing system	SVM use
Mining data	22	5940	5940	Mining industry	Quality level prediction

Dataset	Kernel	σ	C	Accuracy on test set	Accuracy drop
Mining data	RBF	0.364	23.4	82.34%	-0.40%

Étude de robustesse

Robustesse d’un modèle ML

A first objective is not to control the impact of uncertainties on the performance of machine learning techniques, but to :

1. Quantify the impact of these uncertainties on the performance of classifiers

Étude de robustesse

Robustesse d’un modèle ML

A first objective is not to control the impact of uncertainties on the performance of machine learning techniques, but to :

1. Quantify the impact of these uncertainties on the performance of classifiers

2. **Identify the parameters that contribute most to this impact.**

Étude de robustesse

Robustesse d'un modèle ML

A first objective is not to control the impact of uncertainties on the performance of machine learning techniques, but to :

1. Quantify the impact of these uncertainties on the performance of classifiers
2. **Identify the parameters that contribute most to this impact.**

How to ?

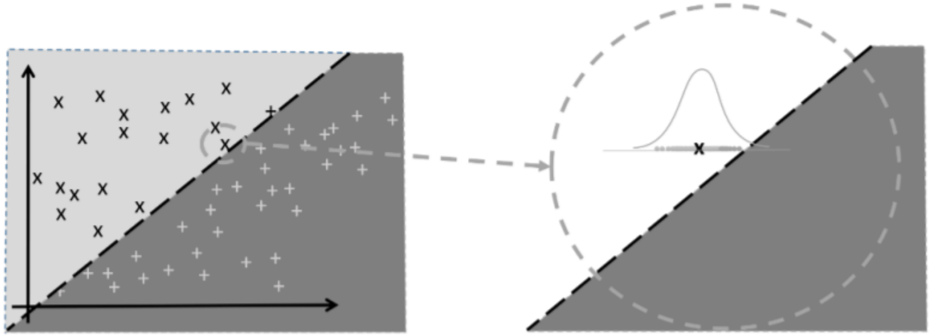
SOBOL sensitivity analysis consisting of dividing the variance of the output into fractions that can be attributed to the inputs.

Identification of key parameters

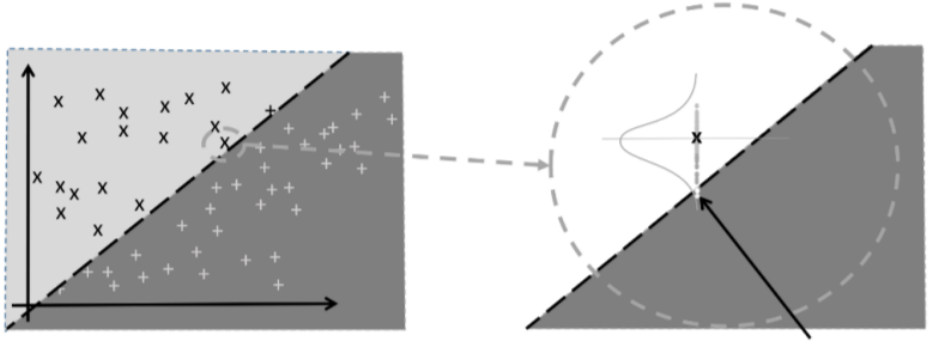
Introduction

The Monte Carlo simulation results do not identify the parameters uncertainties that have the most impact on the SVM accuracy.

In the first figure, it is shown that the uncertainties of the first parameter have no effect on the classification of the data point;



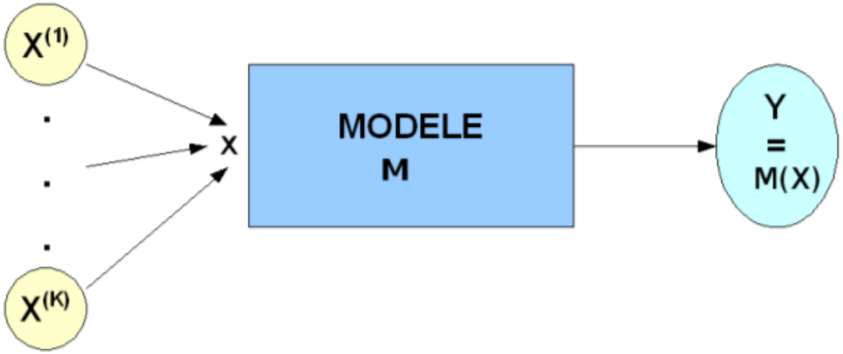
on the other hand, the uncertainties associated with the second parameter can lead to a misclassification of this data point.



The objective of the SOBOL sensitivity analysis is to answer this question and identify the parameters that contribute most to the drop in prediction accuracy.

Identification of key parameters

Sobol analysis (SA)



Aims of SA

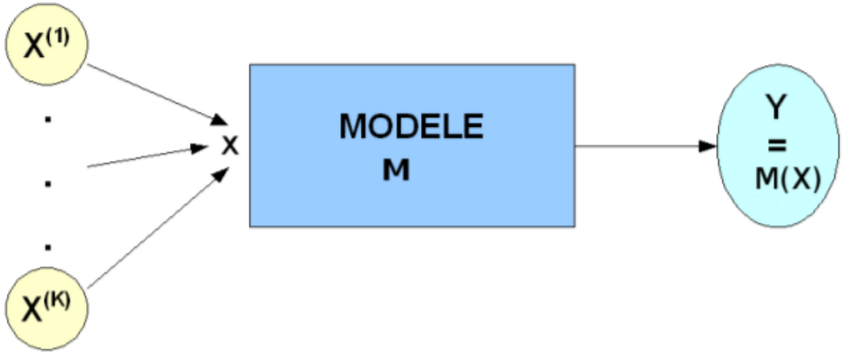
- ***Uncertainty analysis* :** uncertainties, variabilities on $\boldsymbol{X_i}$ ⇒ Effects on $\boldsymbol{Y}$
- ***Sensitivity analysis* :** which $\boldsymbol{X_i}$ are the most influential on $\boldsymbol{Y}$

Model = black box (analytical form may be unknown)

$$\boldsymbol{Y} = \boldsymbol{M}(\boldsymbol{X_1}, \dots, \boldsymbol{X_d})$$

Identification of key parameters

Sobol analysis (SA)



Aims of SA

- ***Uncertainty analysis*** **:** uncertainties, variabilities on X_i ⇒ Effects on Y
- ***Sensitivity analysis*** **:** which X_i are the most influential on Y

Model = black box (analytical form may be unknown)

$$Y = \boldsymbol{M}(X_1, \dots, X_d)$$

Our goal : is to measure the overall influence of the input factors X_i on Y

- over a range of variation of X_i
- with the minimum of assumptions on the model
- by integrating the interactions between the X_i

Identification of key parameters

Sobol indices

Sobol analysis is based on **variance decomposition** techniques to provide a quantitative measure of the contributions of the input to the output variance.

The decomposition of the output variance in a Sobol sensitivity analysis uses the same principle as the classical analysis of variance in a factorial design, allowing the output variance to be represented as follows:

$$V = \sum_{i=1}^{p} V_i + \sum_{1 \le i \le j \le p}^{p} V_{ij} + \cdots + V_{1\ldots p}$$

Where :

V : The total variance of the model output.

$\mathrm{V_i}$: The first order contribution of the $\mathrm{i^{th}}$ parameter of the model.

$\mathrm{V_{ij}}$: The contribution of the interaction of the $\mathrm{i^{th}}$ and $\mathrm{j^{th}}$ parameters.

Identification of key parameters

Sobol indices

Based on this decomposition, Sobol defines:

- First order sensitivity indices

$$S_i = \frac{V_i}{V}$$

A quantitative measure of the contribution of the main effect of the input parameter X_i to the variance of the output.

- Higher order sensitivity indices

$$S_{ij} = \frac{V_{ij}}{V} \; ; \quad S_{ijk} = \frac{V_{ijk}}{V}$$

A quantitative measure of the contribution of the interaction of the $\mathrm{i^{th}}$ and $\mathrm{j^{th}}$ parameters to the variance of the output.

- How to interpret the Sobol indices?

Being all positive, the larger the index (close to 1), the more important the parameter impact is.

Identification of key parameters

Sobol indices

The number of sensitivity indices thus constructed, from order 1 to order p, is equal to $(2^p - 1)$.

When the number of input variables p is too large, the number of sensitivity indices increases dramatically. Estimation and interpretation of all these indices quickly becomes impossible.

To solve this problem, Homma and Saltelli introduced "**Sobol total effect**" indices, which express the total sensitivity of the variance Y to a variable Xi.

The total sensitivity index S_{T_i} is defined as the sum of the contribution caused by X_i and its interactions -of any order- with any other input variable.

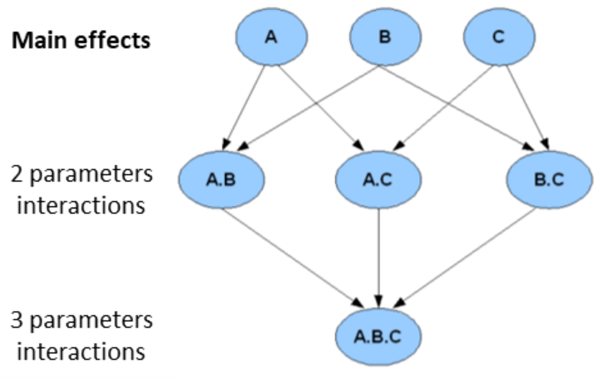
$$S_{T_i} = \sum_{k\#i} S_k$$

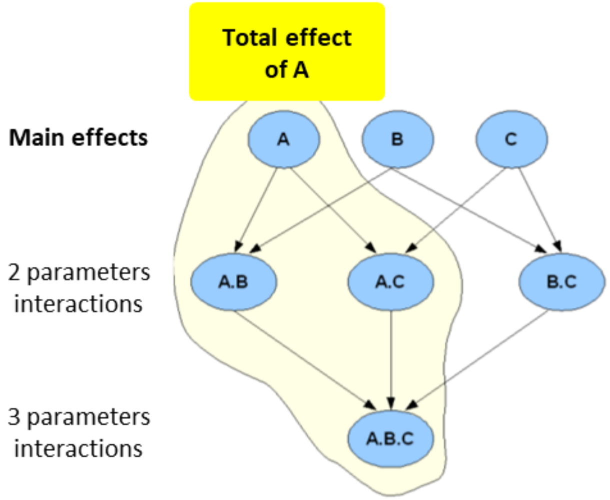
Where #i represents all sets containing index i

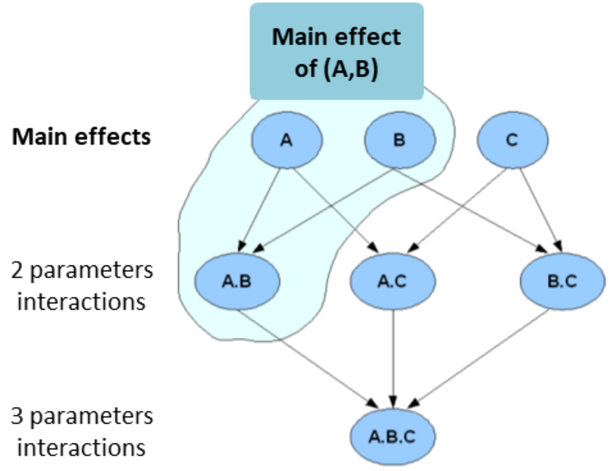
Identification of key parameters

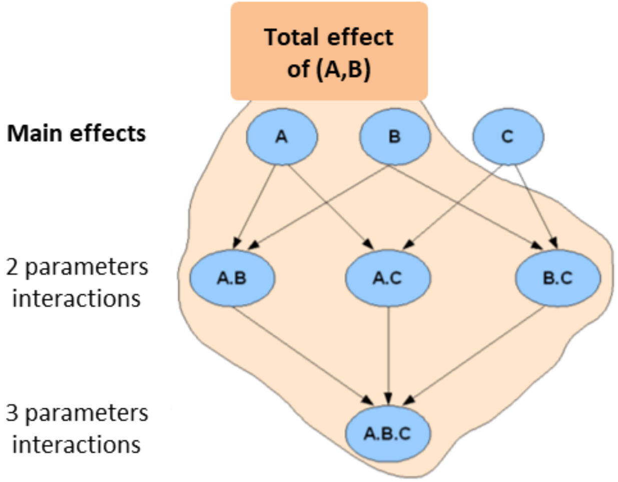
Illustration of the Sobol indices

For a system defined by 3 parameters:









Identification of key parameters

Estimation of the Sobol indices

Exact calculation of Sobol indices is often impossible, since :

- large dimensional integrals are used,
- in machine learning we are faced with models where the analytical form of the model is not known.

One way to estimate the Sobol indices is to apply a sampling approach based on Monte-Carlo simulation.

Identification of key parameters

Estimation of the Sobol indices

Exact calculation of Sobol indices is often impossible, since :

- large dimensional integrals are used,
- in machine learning we are faced with models where the analytical form of the model is not known.

One way to estimate the Sobol indices is to apply a sampling approach based on Monte-Carlo simulation.

Let's take two samples A and B

$$A = \begin{pmatrix} x_{A,1;1} & \cdots & x_{A,1;d} \\ \cdots & \cdots & \cdots \\ x_{A,N;1} & \cdots & x_{A,N;d} \end{pmatrix}$$

$$\rightarrow C_i = \begin{pmatrix} x_{A,1;1} & \cdots & x_{B,1;i} & \cdots & x_{A,1;d} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{A,N;1} & \cdots & x_{B,N;i} & \cdots & x_{A,N;d} \end{pmatrix}$$

$$i = 1, \ldots, d$$

$$B = \begin{pmatrix} x_{B,1;1} & \cdots & x_{B,1;d} \\ \cdots & \cdots & \cdots \\ x_{B,N;1} & \cdots & x_{B,N;d} \end{pmatrix}$$

Identification of key parameters

Estimation of the Sobol indices

Once $\boldsymbol{C_i}$ is defined, we use the model at disposition to compute the different outputs:

$$A = \begin{pmatrix} x_{A,1;1} & \cdots & x_{A,1;i} & \cdots & x_{A,1;d} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{A,N;1} & \cdots & x_{A,N;i} & \cdots & x_{A,N;d} \end{pmatrix} \overset{\mathcal{M}}{\rightarrow} Y_A = \begin{pmatrix} y_{A,1} \\ \vdots \\ y_{A,N} \end{pmatrix}$$

$$C_i = \begin{pmatrix} x_{A,1;1} & \cdots & x_{B,1;i} & \cdots & x_{A,1;d} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{A,N;1} & \cdots & x_{B,N;i} & \cdots & x_{A,N;d} \end{pmatrix} \overset{\mathcal{M}}{\rightarrow} Y_{C_i} = \begin{pmatrix} y_{C_i,1} \\ \vdots \\ y_{C_i,N} \end{pmatrix}$$

$$B = \begin{pmatrix} x_{B,1;1} & \cdots & x_{B,1;i} & \cdots & x_{B,1;d} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{B,N;1} & \cdots & x_{B,N;i} & \cdots & x_{B,N;d} \end{pmatrix} \overset{\mathcal{M}}{\rightarrow} Y_B = \begin{pmatrix} y_{B,1} \\ \vdots \\ y_{B,N} \end{pmatrix}$$

$\Rightarrow$ Estimation of $\boldsymbol{S_{T_i}}$ using $\boldsymbol{A}$ and $\boldsymbol{C_i}$

$\Rightarrow$ Estimation of $\boldsymbol{S_i}$ using $\boldsymbol{B}$ and $\boldsymbol{C_i}$

Identification of key parameters

Estimation of the Sobol indices

Thus, we can estimate the first-order and total Sobol indices by :

$$\hat{S}_i = \frac{\hat{V}_i}{\hat{V}} ,$$

$$\hat{S}_{T_i} = \frac{\hat{V}_{-(i)}}{\hat{V}}$$

where:

$$\hat{V} = \text{Total variance of the output}$$

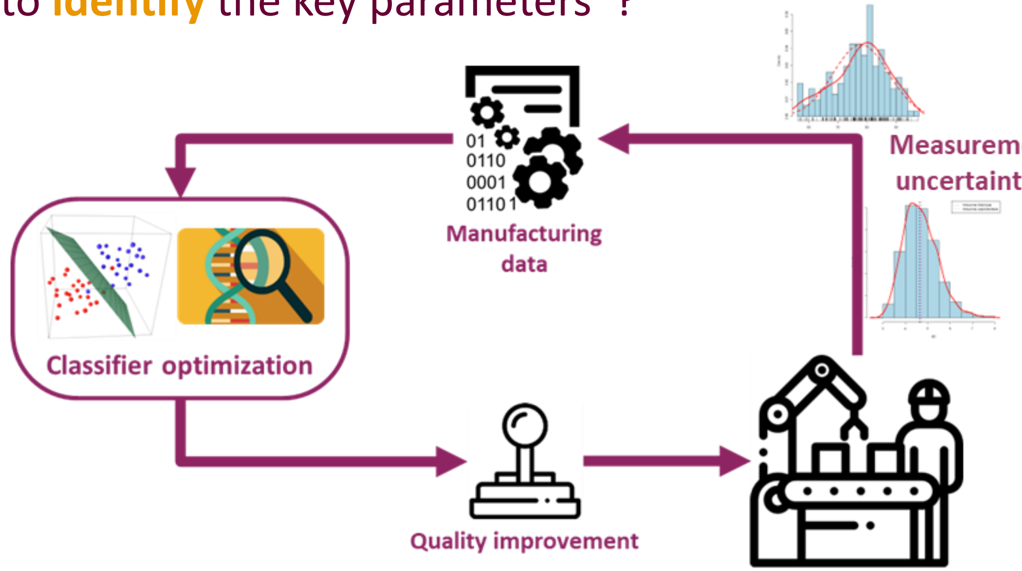
and:

$$\hat{V}_i = \frac{1}{N} \sum_{k=1}^{N} y_{B,k}(y_{C_i,k} - y_{A,k})$$

$$\hat{V}_{-(i)} = \frac{1}{N} \sum_{k=1}^{N} y_{A,k}(y_{A,k} - y_{C_i,k})$$

Project – issue 2

1. How to **quantify** the impact of uncertainties on SVM accuracy ?
2. How to **identify** the key parameters ?



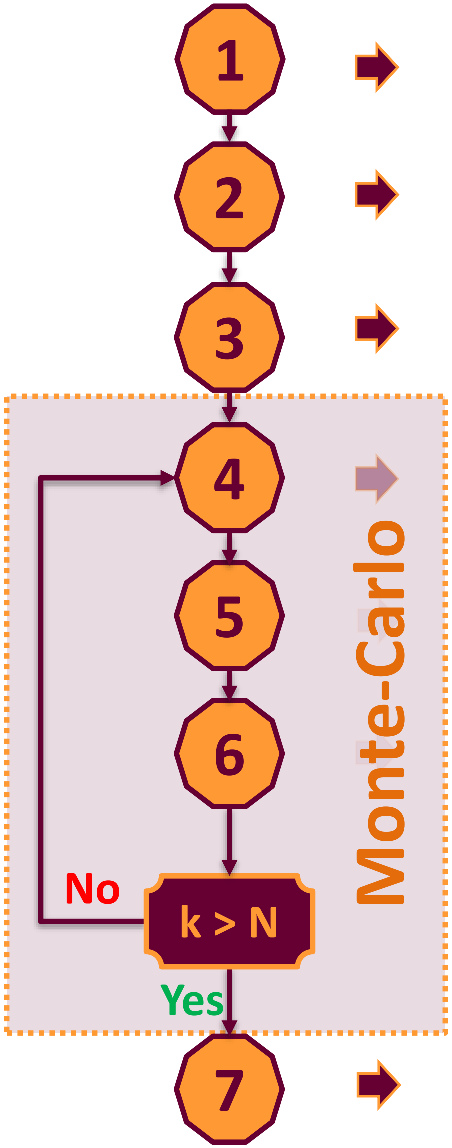
In this project, and by using the previous results found in the 1st part, **you are going to focus on** :

- ✓ The implementation of the Monte-Carlo simulation for the quantification of the uncertainties impact on SVM prediction accuracy.
- ✓ Implementing Sobol sensitivity analysis for estimating Sobol indices, i.e., the uncertainties of the parameters that have the most impact on the accuracy of the SVM.
- ✓ The analysis of the different results.

A step-by-step guide for the implementation of Monte-Carlo and Sobol analysis is provided in the following slides.

Quantifying the impact of uncertainties

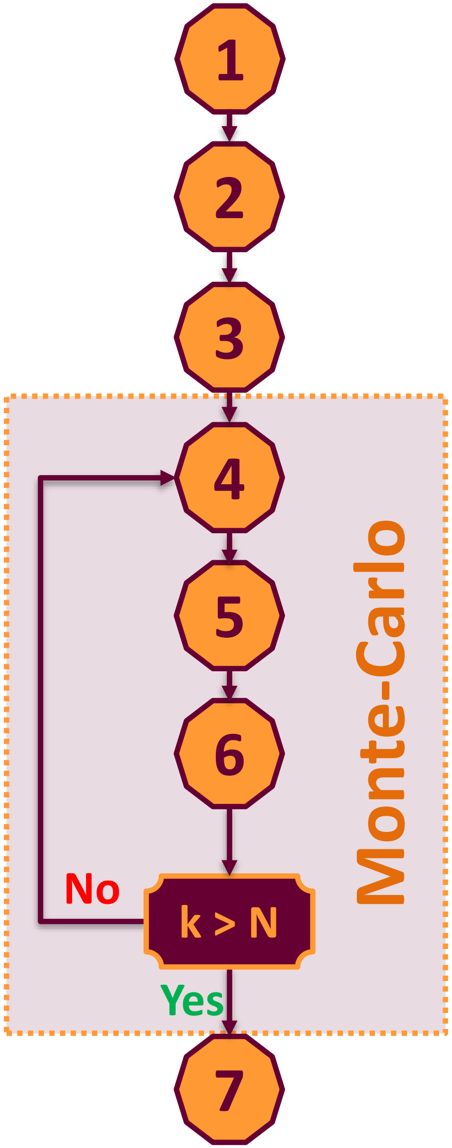
Monte-Carlo for impact quantification



1. Preprocess the dataset : data split, data normalization, …
2. Train the SVM model using the training set.
3. Generate 1000 (**N**) random Gaussian noises.
4. Noise the validation/test set with the k^{th} noise.
5. Assess the prediction accuracy of SVM.
6. Store the found result.
7. Quantify the impact of uncertainties.

Quantifying the impact of uncertainties

Monte-Carlo for impact quantification

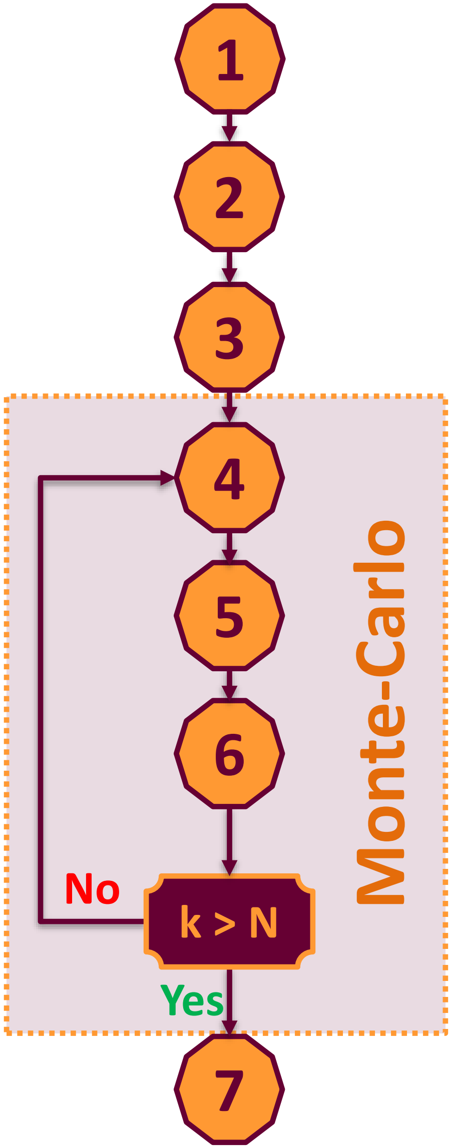


1. **Preprocess the dataset : data split, data normalization, …**

- Import the dataset.
- Normalize (scale) the dataset : using the Min and Max values of the training set.
- Split the dataset into training, validation and test sets.

Quantifying the impact of uncertainties

Monte-Carlo for impact quantification

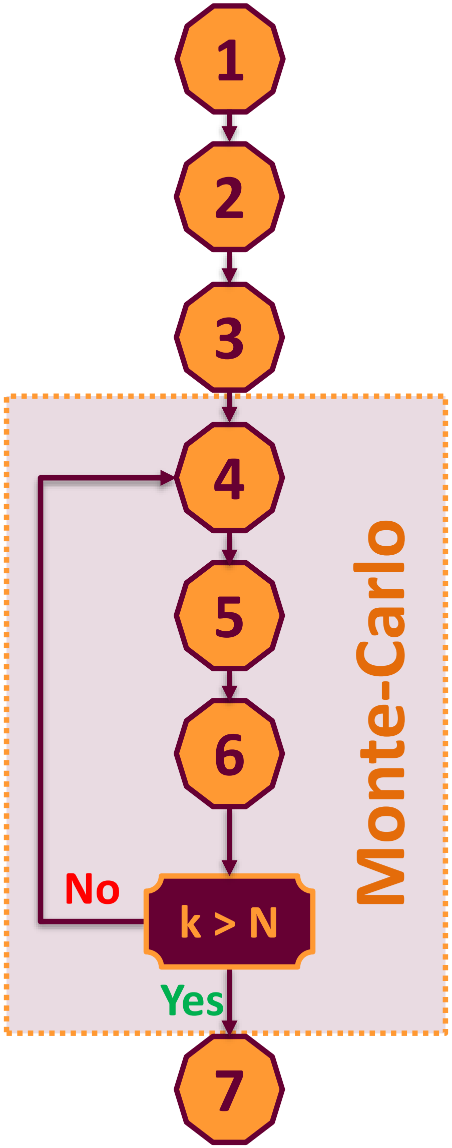


2. **Train the SVM model using the training set.**

   - You have to use the optimized SVM model with the optimal hyperparameters found in the previous lecture.

Quantifying the impact of uncertainties

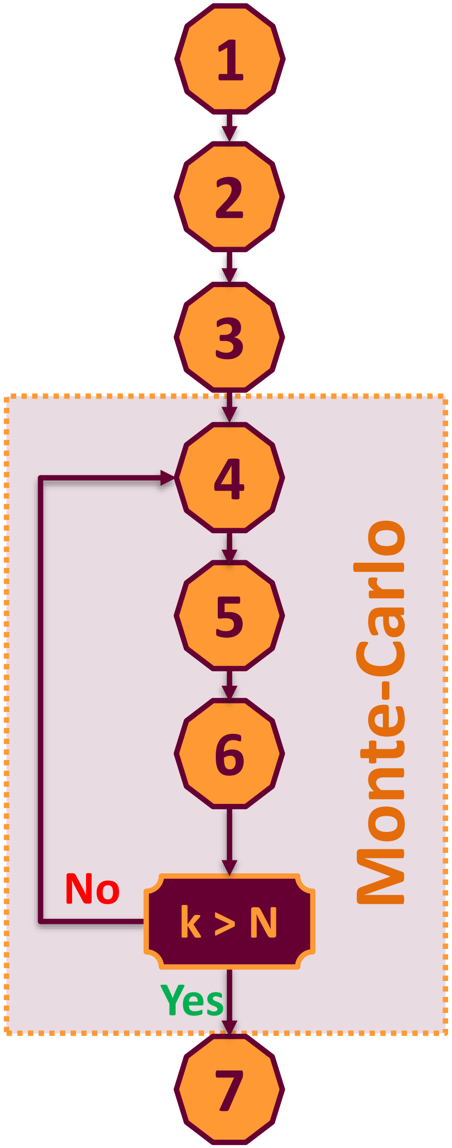
Monte-Carlo for impact quantification



3. **Generate 1000 (N) random Gaussian noises.**

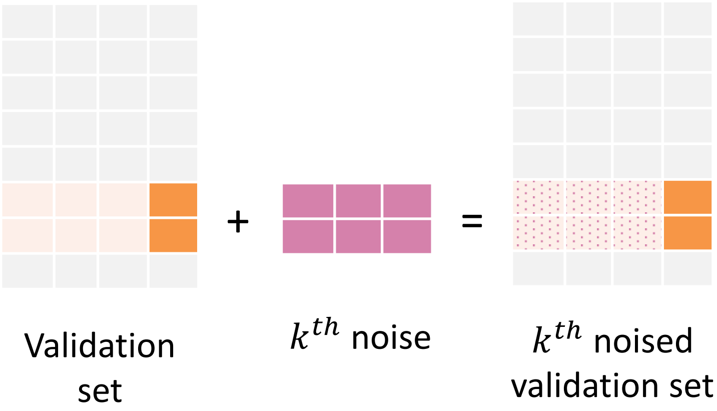
- Calculate the standard deviation $\boldsymbol{\sigma_i}$ of each parameter $\boldsymbol{X_i}$ (*NB : the calculation must be made on the* ***non normalized*** *dataset)*
- Generate 1000 matrices having the same size of the validation set, where each column $\boldsymbol{"i"}$ is generated randomly following the gaussian distribution $\boldsymbol{N(0; \frac{2.5*\sigma_i}{100})}$

Monte-Carlo for impact quantification



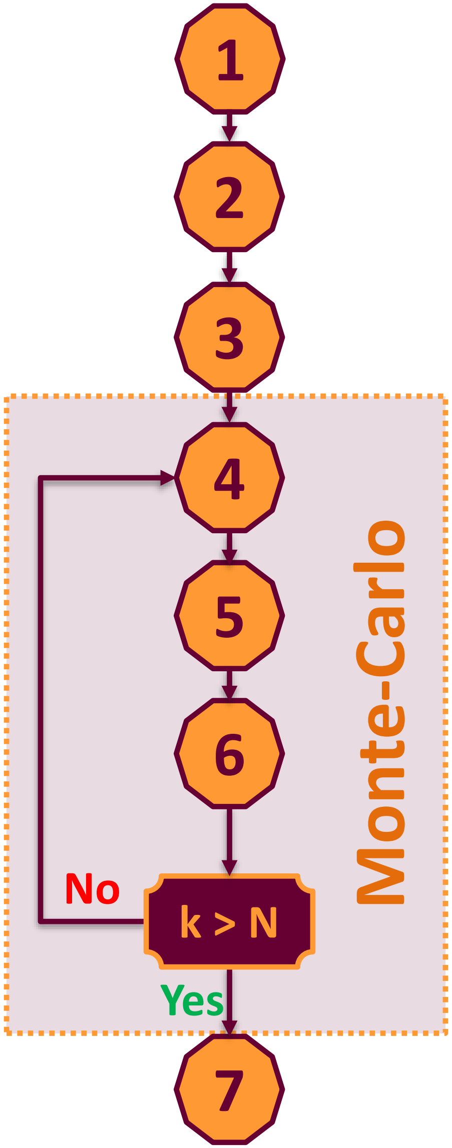
4. **Noise the validation/test set with the k^{th} noise.**

- Add the k^{th} randomly generated matrice of noise to the validation set, with k in range(0,1000).



Quantifying the impact of uncertainties

Monte-Carlo for impact quantification

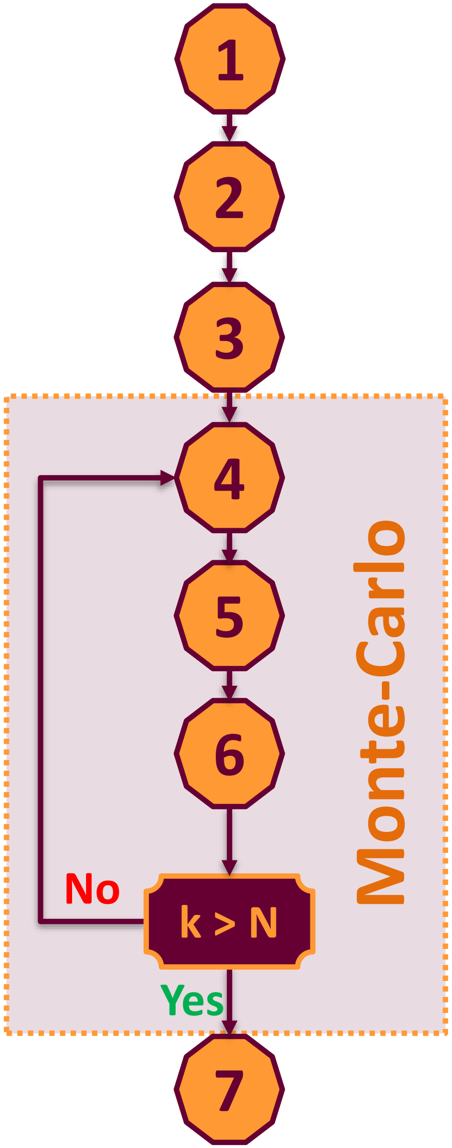


5. **Assess the prediction accuracy of SVM.**

- Normalize (scale) the $\boldsymbol{k^{th}}$ **noised validation set** : using the Min and Max values of the **training set**.
- By using the SVM model trained in step 2, calculate the prediction accuray of the $\boldsymbol{k^{th}}$ **noised validation set.**
- Calculate the **drop of accuracy** defined as the difference between the prediction accuracy of the validation set and the prediction accuracy of the k^{th} noised validation set.

Quantifying the impact of uncertainties

Monte-Carlo for impact quantification

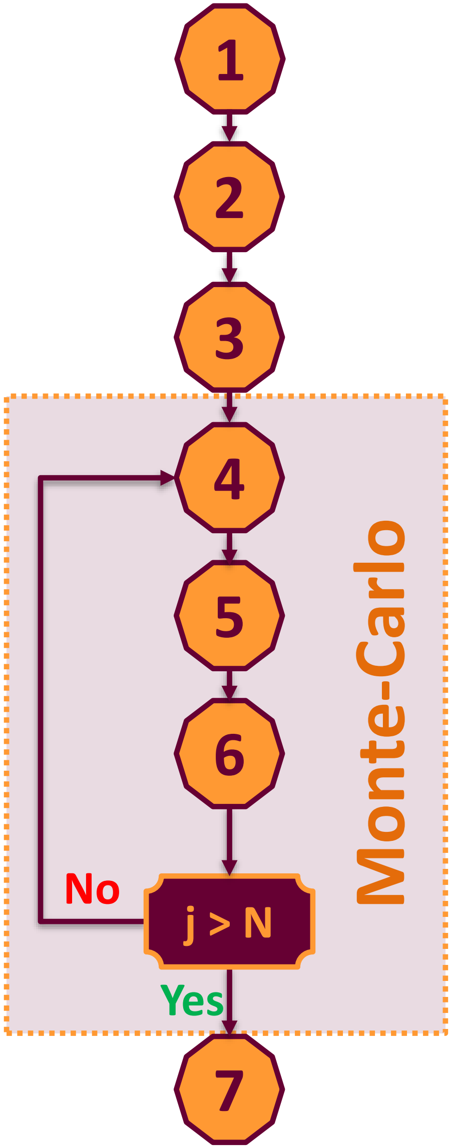


6. **Store the found result**

- Store the drop of accuracy somewhere.
- Check if k > N.

Quantifying the impact of uncertainties

Monte-Carlo for impact quantification



7. **Quantify the impact of uncertainties.**

   - Calculate the mean of the drops of accuracy.

Identifying the key parameters

Sobol indices

Goal :

Estimating Sobol indices $\Longrightarrow$ key parameters identification, i.e., identification of parameters' uncertainties with the most impacts.

Principle of Sobol analysis :

Dividing the output variance into fractions that can be attributed to the inputs.

How to ? :

One of the ways to estimate Sobol indices is by using the Monte-Carlo simulation. The Monte-Carlo estimation consists of estimating:

1. the output expected value $(E[Y])$,
2. the output variance $(V[Y])$,
3. and two quantities $(\hat{U}_i)$ and $(\hat{U}_{\sim i})$.

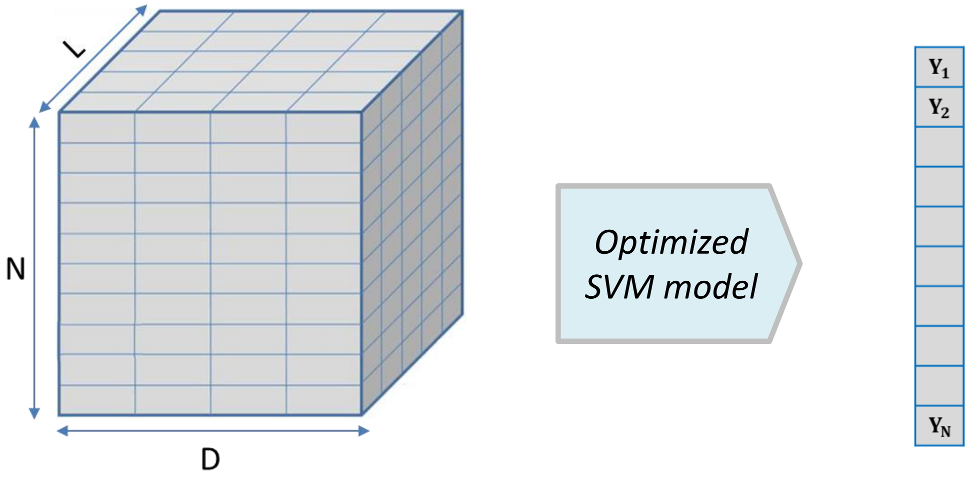
The estimation of these **four quantities** requires **two N-samples**.

Identifying the key parameters

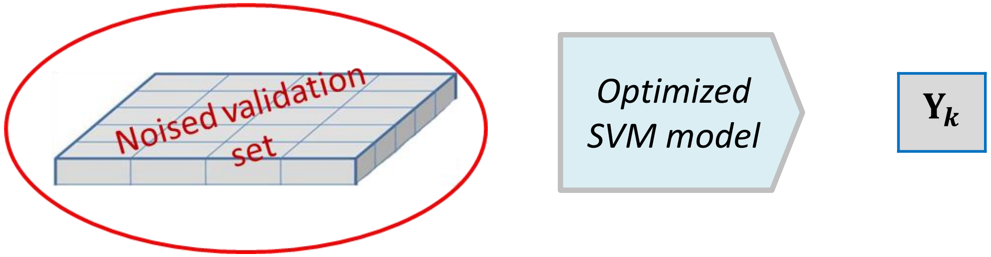
Step 0

In this step, the definition of the N-sample and of the output is given.

The N-sample in our case is : 1000 noised validation sets.



L : Number of data points.
D : Number of parameters.
N : Sample size.



Each Y_k is calculated from a data matrix (*noised validation set*) $(Lxd)_k$.

Y_k = (SVM accuracy of **initial validation set**) – (SVM accuracy of the k^{th} **noised validation set)**

Identifying the key parameters

Step 1 – Expected value

Formula :

$$\hat{\mathrm{E}} = E[Y] = \frac{1}{N}\sum_{k=1}^{N} f(x_{k1}, \dots, x_{kp})$$

Interpretation :

In other words, you need to calculate the mean of all the $\boldsymbol{Y_k}$.

Good news !! You've already performed this in the first part of this project.

Identifying the key parameters

Step 2 – Variance

Formula :

$$\hat{V} = V[Y] = \frac{1}{N}\sum_{k=1}^{N} f^2(x_{k1}, \dots, x_{kp}) \; - \; \hat{\mathrm{E}}^2$$

Interpretation :

The variance can be formulated as a function of the expected value.

It is the difference between the sum of the squares of $\boldsymbol{Y_k}$ and the square of the expected value.

$$\hat{V} = \frac{1}{N}\sum_{k=1}^{N} Y_k^2 \; - \; \widehat{E^2}$$

Identifying the key parameters

Step 3 – First order Sobol indices

The estimation of the first order Sobol sensitivity indices consists in estimating the quantity $\boldsymbol{V_i}$

$$V_i = \mathrm{V}(\mathrm{E}[Y|X_i]) = \underbrace{\mathrm{E}[\mathrm{E}[Y|X_i]^2]}_{U_i} - \mathrm{E}[\mathrm{E}[Y|X_i]]^2 = U_i - \mathrm{E}[Y]^2$$

Sobol proposes to estimate the quantity $\boldsymbol{U_i}$ like a classical expected value but taking into account the conditionality to $\boldsymbol{X_i}$ by varying between the two calls to the function $\boldsymbol{f}$ (*trained SVM model in our case*) all the variables except the variable $\boldsymbol{X_i}$. This requires two samples of the input variables.

$$\hat{\mathrm{U}}_i = \frac{1}{N}\sum_{k=1}^{N} f(x_{k1}^{(1)}, \ldots, x_{k(i-1)}^{(1)}, x_{ki}^{(1)}, x_{k(i+1)}^{(1)}, \ldots, x_{kp}^{(1)}).f(x_{k1}^{(2)}, \ldots, x_{k(i-1)}^{(2)}, x_{ki}^{(1)}, x_{k(i+1)}^{(2)}, \ldots, x_{kp}^{(2)})$$

First order sensitivity indices can thus be calculated :

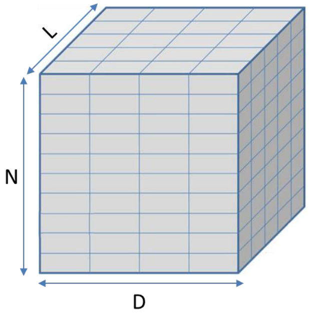
$$\hat{S}_i = \frac{\hat{\mathrm{U}}_i - \hat{\mathrm{E}}^2}{\hat{V}}$$

Identifying the key parameters

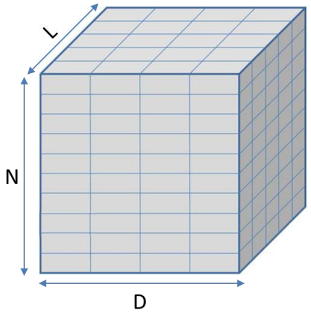
Step 3 – First order Sobol indices

"Let's get practical and calculate the first order Sobol index of the first parameter X_1 !!"

1. Generate two 1000-samples of noised validation sets.

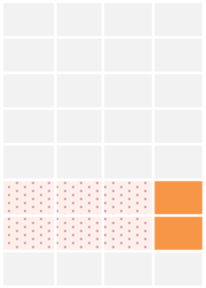


Sample A

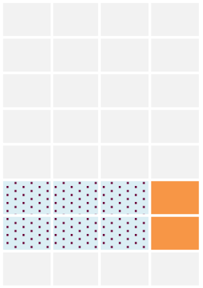


Sample B

2. Pick the $\boldsymbol{k^{th}}$ noised validation set from each sample.
3. Create a third noised validation set from the two picked validation sets

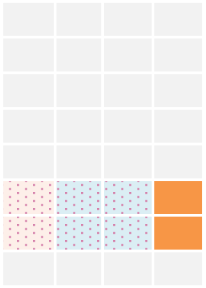


$\boldsymbol{A_k}$: k^{th} noised validation set from **sample A**



$\boldsymbol{B_k}$: k^{th} noised validation set from **sample B**

Inject all the parameters of $\boldsymbol{B_k}$ into $\boldsymbol{A_k}$, except the first parameter $\boldsymbol{X_1}$.

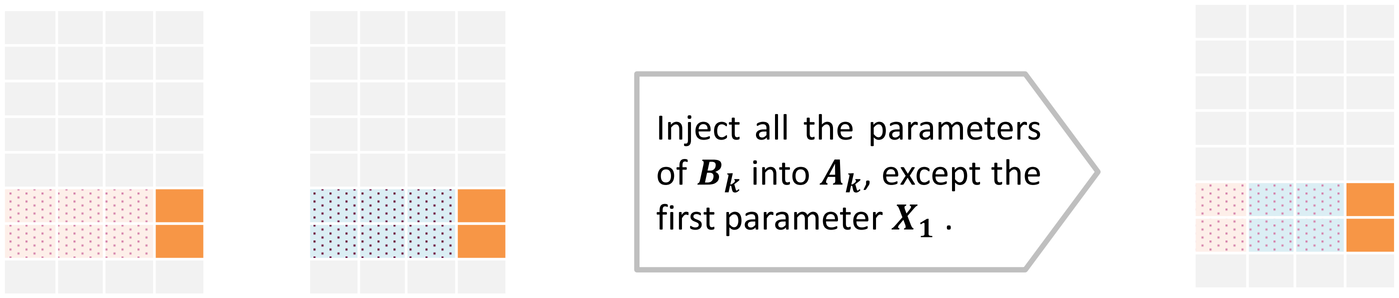


$\boldsymbol{C_k}$: k^{th} generated noised validation set

Identifying the key parameters

Step 3 – First order Sobol indices

2. Pick the $\boldsymbol{k^{th}}$ noised validation set from each sample.
3. Create a third noised validation set from the two picked validation sets



Inject all the parameters of $\boldsymbol{B_k}$ into $\boldsymbol{A_k}$, except the first parameter $\boldsymbol{X_1}$.

$\boldsymbol{A_k}$: k^{th} noised validation set from sample A

$\boldsymbol{B_k}$: k^{th} noised validation set from sample B

$\boldsymbol{C_k}$: k^{th} generated noised validation set

4. Predict $\boldsymbol{A_k}$ $(= \boldsymbol{Y_{A_k}})$ then $\boldsymbol{C_k}$ $(= \boldsymbol{Y_{C_k}})$ using the optimized SVM model.
5. Multiply both prediction accuracies : $\boldsymbol{Y_{A_k}} * \boldsymbol{Y_{C_k}}$
6. Repeat steps 2, 3, 4 and 5 for k in range(0,1000).
7. Calculate the first order Sobol index of the first parameter X_1.

$$\widehat{\boldsymbol{S_1}} = \frac{(\frac{1}{N}\sum_{k=1}^{N} \boldsymbol{Y_{A_k}} * \boldsymbol{Y_{C_k}}) - \widehat{E^2}}{\hat{V}}$$

8. Do the same previous steps for all the parameters X_i to calculate all first order Sobol indices.

Identifying the key parameters

Step 4 – Sobol total effect indices

The Sobol total effect indices can be estimated directly as follow :

$$S_{T_i} = 1 - \frac{\mathrm{V}(\mathrm{E}[Y|X_{\sim i}])}{\mathrm{V}(Y)} = 1 - \frac{V_{\sim i}}{V}$$

Where $\boldsymbol{V_{\sim i}}$ is the variance of the expected value of $\boldsymbol{Y}$ conditional on all variables except $\boldsymbol{X_i}$. $\boldsymbol{V_{\sim i}}$ is then estimated as $\boldsymbol{V_i}$, except that instead of varying all the variables except $\boldsymbol{X_i}$, we only vary only $\boldsymbol{X_i}$.

$$V_{\sim i} = \mathrm{E}[\mathrm{E}[Y|X_{\sim i}]^2] - \mathrm{E}[\mathrm{E}[Y|X_{\sim i}]]^2 = U_{\sim i} - \mathrm{E}[Y]^2$$

and :

$$\hat{U}_{\sim i} = \frac{1}{N}\sum_{k=1}^{N} f\left(x_{k1}^{(1)}, \ldots, x_{k(i-1)}^{(1)}, x_{ki}^{(1)}, x_{k(i+1)}^{(1)}, \ldots, x_{kp}^{(1)}\right) f\left(x_{k1}^{(1)}, \ldots, x_{k(i-1)}^{(1)}, x_{ki}^{(2)}, x_{k(i+1)}^{(1)} \ldots, x_{kp}^{(1)},\right)$$

Sobol Total-effect indices indices can therefore be calculated :

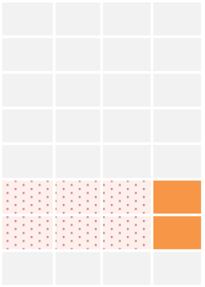
$$\hat{S}_{Ti} = 1 - \frac{\hat{\mathrm{U}}_{\sim i} - \hat{\mathrm{E}}^2}{\hat{V}}$$

Identifying the key parameters

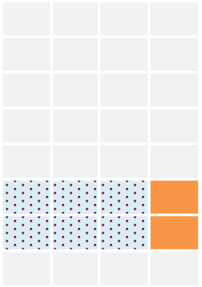
Step 4 – Sobol total effect indices

" We'll focus on the calculation of the Sobol total effect index of the first parameter X_1 !!"

1. Generate two 1000-samples of noised validation sets *(same as in step 3).*
2. Pick the $\boldsymbol{k^{th}}$ noised validation set from each sample *(same as in step 3).*
3. Create a third noised validation set from the two picked validation sets

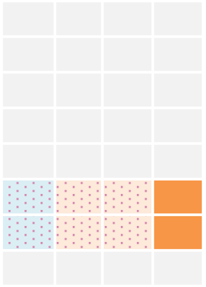


$\boldsymbol{A_k}$: k^{th} noised validation set from **sample A**



$\boldsymbol{B_k}$: k^{th} noised validation set from **sample B**

Inject the first parameter of $\boldsymbol{B_k}$ into $\boldsymbol{A_k}$.



$\boldsymbol{C_k}$: k^{th} created noised validation set

4. Predict $\boldsymbol{A_k}$ $(= \boldsymbol{Y_{A_k}})$ then $\boldsymbol{C_i}$ $(= \boldsymbol{Y_{C_k}})$ using the optimized SVM model.
5. Multiply both prediction accuracies : $\boldsymbol{Y_{A_k}} * \boldsymbol{Y_{C_k}}$
6. Repeat steps 2, 3, 4 and 5 for k in range(0,1000).
7. Calculate the Sobol total effect index the first parameter X_1.

$$\widehat{S_{T_1}} = 1 - \frac{(\frac{1}{N}\sum_{k=1}^{N} \boldsymbol{Y_{A_k}} * \boldsymbol{Y_{C_k}}) - \widehat{E^2}}{\hat{V}}$$

8. Do the same previous steps for all the parameters X_i to calculate all Sobol total effect indices.

Plan

Learning outcomes:

- **How to evaluate and handle the impact of noise (uncertainties) in manufacturing data on the performance of classification techniques [SVM as a case study]?**

Keywords : *Support Vector Machine, Measurment uncertainties, Genetic algorithm, Monte-Carlo simulation, Sobol sensitivity analysis, Robust optimisation.*

1st issue :

- **Objective :** How to **optimise** the prediction accuracy of SVM ?

⟶ Introduction to SVM.

⟶ A genetic algorithm for SVM prediction accuracy optimisation.

2nd issue :

- **Objective :** How to **evaluate** the impact of uncertainties on SVM accuracy ?

⟶ Monte-Carlo simulation for uncertainties impact quantification.

⟶ Sobol sensitivity analysis for key uncertainties identification.

3rd issue :

- **Objective :** How to **mitigate** the impact of uncertainties on SVM accuracy ?

⟶ Bibliographical study.