Statistical Learning course

– Group project report –

A tutorial on support vector regression

Annabelle Lichou, Oscar Loriou, Louis Mayonove, Aimine Meddeb

December 7th, 2025

Contents

List of Abbreviations and Symbols

Notations

$|x|$ Absolute value of $x \in \mathbb{R}$

$[\![a, b]\!]$ Set of integer numbers from $a \in \mathbb{N}$ to $b \in \mathbb{N}$ both included.

$\|x\|$ Euclidean norm of $x \in \mathbb{R}^p$

$\mathbb{R}^p$ Set of column vectors having p lines and whose elements belong to $\mathbb{R}$ (with $p \in \mathbb{N}$). Basically here we use $\mathbb{R}^p$ to dentote $\mathbb{R}^{p,1}$

$\mathbb{R}^{n \times m}$ Set of matrices with n lines and p columns and whose elements belong to $\mathbb{R}$ *(with both $n \in \mathbb{N}$ and $p \in \mathbb{N}$).*

x_i If x is a vector: $x \in \mathbb{R}^p$ then x_i with $i \in [\![1, p]\!]$, denotes its i-th element.

Symbols

$\mathbb{N}$ Set of all positive integer numbers

$\mathbb{R}$ Set of all real numbers

$\mathbb{R}_+$ Set of all positive or null real numbers

Introduction

This work is based on the study of the article "*A tutorial on support vector regression*" by Alex J. Smola and Bernhard Schölkopf [1]. It deals with the use of support vector machines for regression.

The general set up for regression is that we assume we have a training dataset made of $n \in \mathbb{N}$ datapoints:

$$\mathcal{D} = ((x_i, y_i))_{i \in [\![1,n]\!]}$$

with $(x_i, y_i) \in \mathbb{R}^p \times \mathbb{R}$ for all $i \in [\![1, n]\!]$, meaning that each datapoint of the dataset $\mathcal{D}$ has p features which are all real numbers and the target value we want to predict y_i is a real value.

We are looking for a function $f : \mathbb{R}^p \to \mathbb{R}$ which allows to predict the values y_i as good as possible. That is, for all $i \in [\![1, n]\!]$ we want that the $f(x_i)$ be as closed as possible to the y_i's.

- The measure of "how close" is a prediction $f(x_i)$ to the "desired" prediction y_i is measured by a loss function.
- The set of functions among which we look for the "best" prediction function is limited to a given set $\mathcal{F}$ to simplify the problem.

Both these parameters will have influence on the algorithm we can use to find out the best prediction function.

At first the basic idea of Support Vector Regression will be explained through the comparison of the "hard border" and "soft border" regressions of linear problems. Then, its extension to non-linear problems will be explored by using kernels function. In addition, the different cost functions one can use will be stated and compared. Finally, it will be showed that SVR are a special case of Regularization Networks which benefits from its robust convergence properties and overcome its usually long computation time.

All simulations and numerical applications can be found in the following GitHub repository :

`https://github.com/Squenger/SV-Machines.git`

1 Support vector machine for regression in the linear case

1.1 Support vector machine for regression with "hard" borders

1.1.1 Model

Support vector machines for regression with "hard" borders, introduced by Vapnik in 1995, consist in looking for a linear function to predict the targets y_i from the features x_i. That is, we look for a function $f_{\omega,b} : \mathbb{R}^p \to \mathbb{R}$ of the form

$$f_{\omega,b} : \left\{ \begin{array}{ccc} \mathbb{R}^p & \to & \mathbb{R} \\ x_i & \mapsto & f_{\omega,b}(x_i) = \langle \omega, x_i \rangle + b \end{array} \right.$$

where $\omega \in \mathbb{R}^p$ and $b \in \mathbb{R}$ are parameters[1].

The set of functions $\mathcal{F}$ among which we look for the "best" prediction function is thus the same as in the "classic" ordinary least squares regression set up:

$$\mathcal{F} = \left\{ f_{\omega,b} : \left\{ \begin{array}{ccc} \mathbb{R}^p & \to & \mathbb{R} \\ x_i & \mapsto & f_{\omega,b}(x_i) = \langle \omega, x_i \rangle + b \end{array} \right. \quad \middle| \quad \omega \in \mathbb{R}^p,\, b \in \mathbb{R} \right\}$$

However, we use a different criterion to find the "best" prediction function among the set $\mathcal{F}$ compared to the linear regression set up and we also add the constraint that all y_is should be in a range $[-\varepsilon, \varepsilon]$ of the predicted value $f(x_i)$. Thus, the problem to solve is the following:

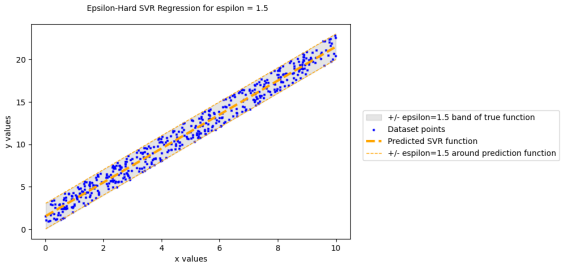
$$\left\{ \begin{array}{ll} \underset{\omega \in \mathbb{R}^p,\, b \in \mathbb{R}}{\operatorname{argmin}} & \frac{1}{2} \|\omega\|^2 \\ s.t. & \forall i \in [\![1, n]\!], \quad |\underbrace{\langle \omega | x_i \rangle + b}_{f(x_i)} - y_i| \leq \varepsilon \end{array} \right.$$

which is equivalent to:

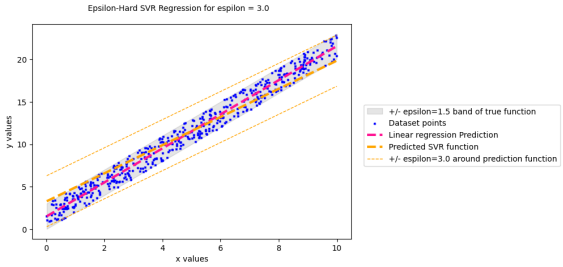
$$\left\{ \begin{array}{ll} \underset{\omega \in \mathbb{R}^p,\, b \in \mathbb{R}}{\operatorname{argmin}} & \frac{1}{2} \|\omega\|^2 \\ s.t. & \forall i \in [\![1, n]\!], \quad \left\{ \begin{array}{lcl} \langle \omega | x_i \rangle + b - y_i & \leq & \varepsilon \\ y_i - \langle \omega | x_i \rangle - b & \leq & \varepsilon \end{array} \right. \end{array} \right. \tag{1}$$

Unlike the linear regression which aims at limiting the sum of "errors" around the predicted function, the support vector machine regression simply enforces that the datapoints are not "too far" from the predicted function *(at maximum ε "far" from it)* but when several linear functions meet this criterion, it chooses the "flattest one", that is the one for which $\|\omega\|$ is the lowest, not the one limiting the sum of errors (see 1).

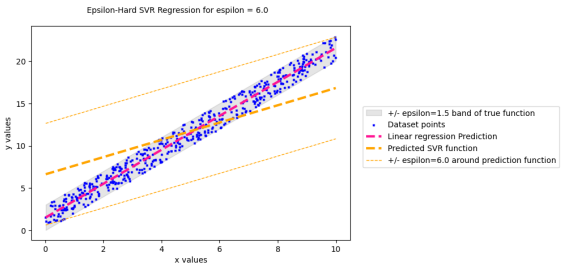
[1] $\mathcal{F}$ is a set of parametric functions.



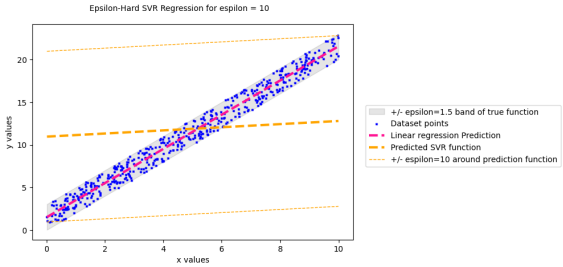
(a) True ε of the dataset is 1.5 and we set it exactly 1.5 in the resolution of 1.



(b) True ε of the dataset is 1.5, but set it to 3 in the resolution of 1.



(c) True ε of the dataset is 1.5 and we set it exactly 6 in the resolution of 1.



(d) True ε of the dataset is 1.5, but set it to 10 in the resolution of 1.

Figure 1: Support vector machine regression obtained for the same dataset but different choices of ε in 1.

1.1.2 Limitations

One severe limitation is that problem 1 has no solution if the points of the dataset are not truly distributed around a linear function within a range $[-\varepsilon, \varepsilon]$ or if there are outliers. Hence, for the dataset of figure 2 and a ε value set to half the height of the grey band. While for the dataset of figure 2, there is not solution to problem 1 for the same ε value.

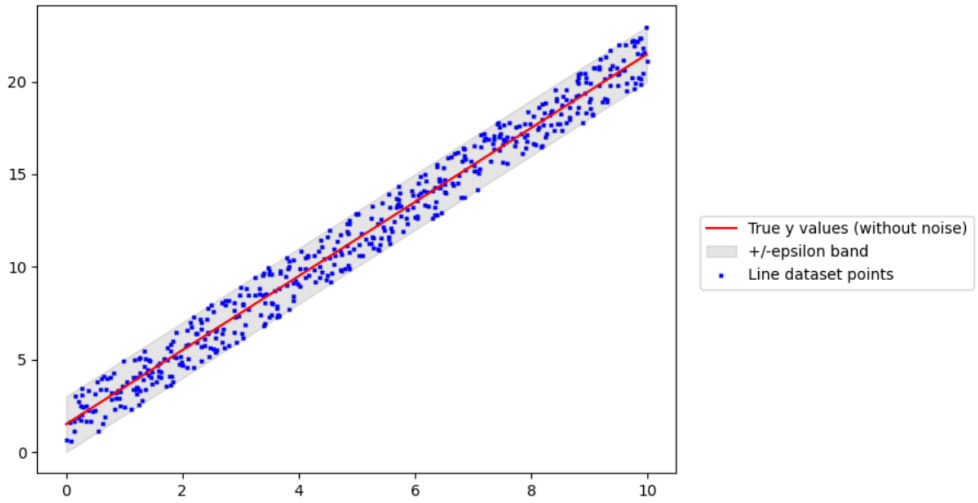


Figure 2: Dataset for which Problem 1 has a solution for ε set to half the height of the light grey border.

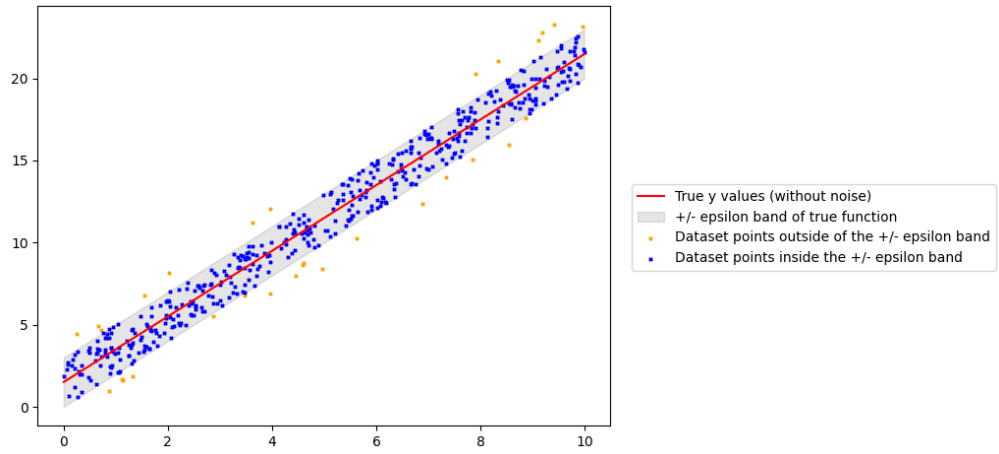


Figure 3: Dataset for which Problem 1 does not have a solution for ε set to half the height of the light grey border.

> We have implemented a "hard" border version of the support vector machine regression in the `epsilon_sv_regression_hard_border` module using the `cvxpy` third party library to solve convex optimization problems.

Such limitation can be overcomed only by setting the value ε used in Equation 1 larger until it "wraps" the dataset, but as shown in Figures 1, the prediction function outputted won't be good if ε is chosen too large.

1.2 Support vector machines for regression with soft borders

1.2.1 Model

One solution to the limitations of the hard border version of support vector machines is to relax the Problem 1 by allowing some of the datapoints not to be inside the border $[-\varepsilon, \varepsilon]$ but penalize them in the cost function. This is done by adding **slack variables** to the problem to account for datapoints outside of the desired ε-border.

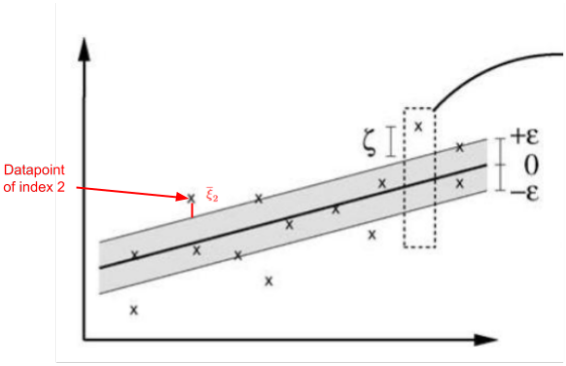
$$\begin{cases} \underset{\omega \in \mathbb{R}^p,\, b \in \mathbb{R}}{\operatorname{argmin}} & \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{n}(\underline{\xi}_i + \overline{\xi}_i) \\ s.t. \quad \forall i \in [\![1, n]\!], & \begin{cases} \langle \omega | x_i \rangle + b - y_i & \leq & \varepsilon + \overline{\xi}_i \\ y_i - \langle \omega | x_i \rangle - b & \leq & \varepsilon + \underline{\xi}_i \\ \overline{\xi}_i \geq 0 \\ \underline{\xi}_i \geq 0 \end{cases} \end{cases} \tag{2}$$

Problem 1 has been extended here by adding:

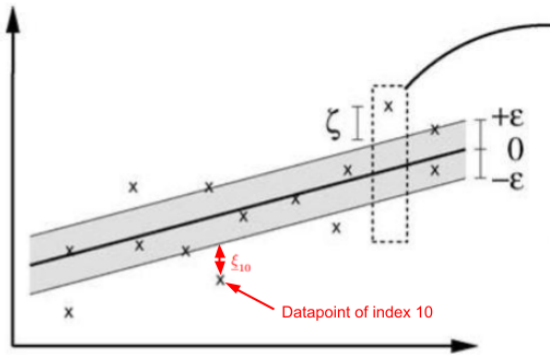
- $2n$ slack variables *(2 for each point of the dataset)* which represent the "extra" distance of the datapoint each side of the ε-border of the prediction function *(if the datapoint is not within the border)*.
- $2n$ constraints which enforce the positivity of all of the slack variables.
- An additive cost to the cost function which is used to pick the best prediction function within $\mathcal{F}$, which penalizes the sum of the distances to the border for all points that are not within the desired border.

The slack variables:

- Each of the variables of the family $\left(\overline{\xi}_i\right)_{i\in[\![1,n]\!]}$ represent the "how far" the value y_i of the datapoint of the corresponding index in the dataset is from the top of the ε-border around the prediction function if it is above that border (otherwise it is null).
- Each of the variables of the family $\left(\underline{\xi}_i\right)_{i\in[\![1,n]\!]}$ represent the "how far" the value y_i of the datapoint of the corresponding index in the dataset is from the bottom of the ε-border around the prediction function if it is below that border (otherwise it is null).



(a) Example of datapoint for which the slack variable $\overline{\xi}_i$ is not null.



(b) Example of datapoint for which the slack variable $\underline{\xi}_i$ is not null.

Note that problem 2 is equivalent a ridge regression with the ε-sensitive loss function instead of the square loss.

$$\underset{\omega\in\mathbb{R}^p,\, b\in\mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{n} \ell(\underbrace{\langle\omega, x_i\rangle + b}_{f(x_i)},\, y_i) + \underbrace{\frac{1}{2}}_{\lambda}\underbrace{\|\omega\|^2}_{\text{pen}(\omega)}$$

where

$$\ell : \begin{cases} \mathbb{R}^2 & \to & \mathbb{R} \\ (f(x_i), y_i) & \mapsto & |f(x_i) - y_i|_\varepsilon = \max\left(\{0\,,\, |f(x_i) - y_i| - \varepsilon\}\right) \end{cases}$$

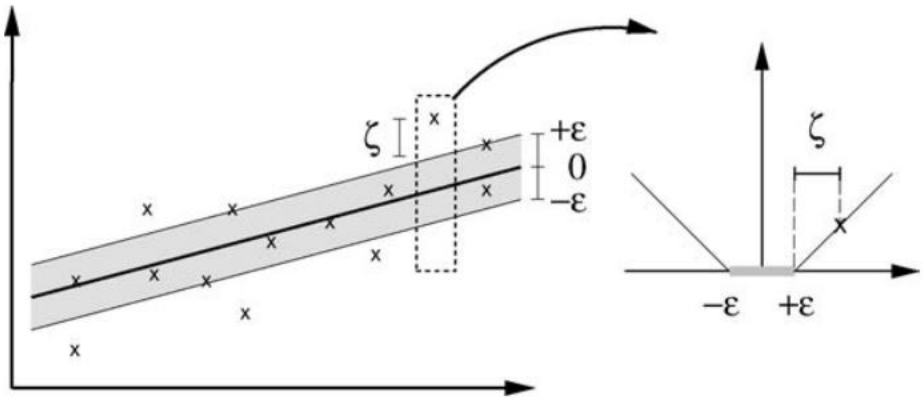


Figure 5: The epsilon sensitive loss *(taken from [1])*.

1.2.2 Solving the soft border problem

In Problem 2:

- The function to optimize is strictly convex as a sum of:
  - the squared norm of the vector ω which is a strictly convex function.
  - A linear combination of the slack variables, which is a convex function.
- All constraints are affine constraints.

It is thus a "relatively easy" problem to solve as we have many results for optimization of problems of this form. We can indeed use the Lagrangian multipliers method.

The **Lagrangian** associated with the problem 2 is the function

$$\mathcal{L} : \begin{cases} \mathbb{R}^p \times \mathbb{R} \times \mathbb{R}_+^n \times \mathbb{R}_+^n \times \mathbb{R}_+^n \times \mathbb{R}_+^n \times \mathbb{R}_+^n \times \mathbb{R}_+^n & \to & \mathbb{R} \\ (\omega, b, \underline{\xi}, \overline{\xi}, \alpha, \alpha^*, \eta, \eta^*) & \mapsto & \underbrace{\frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{n}(\underline{\xi}_i + \overline{\xi}_i)}_{\text{Function to minimize}} \\ & & -\sum_{i=1}^{n}\alpha_i \underbrace{\left(\varepsilon + \underline{\xi}_i - \langle\omega|x_i\rangle - b + y_i\right)}_{\substack{\text{Inequality constraints corresponding} \\ \text{to the "bottom" of the } \varepsilon \text{ border}}} \\ & & -\sum_{i=1}^{n}\alpha_i^* \underbrace{\left(\varepsilon + \overline{\xi}_i + \langle\omega|x_i\rangle + b - y_i\right)}_{\substack{\text{Inequality constraints corresponding} \\ \text{to the "top" of the } \varepsilon \text{ border}}} \\ & & -\sum_{i=1}^{n} \underbrace{\eta_i\underline{\xi}_i + \eta_i^*\overline{\xi}_i}_{\substack{\text{Inequality constraints} \\ \text{to enforce positivity of the } \xi_i s}} \end{cases} \quad (3)$$

The variables $\alpha, \alpha^*, \eta, \eta^*$ are called the **lagrange multipliers**.

From convex problems optimization theory, for our problem, we know that (see Appendix A.1):

- **Existence and unicity of the solution:** The minimization problem 2 has a solution[2] and there is a unique point $(\hat{\omega}, \hat{b})$ which achieves that minimum.[3]
- **Nescessary and sufficient condition for a point to be a solution:**

$$(\hat{\omega}, \hat{b}, \hat{\underline{\xi}}, \hat{\overline{\xi}}, \hat{\alpha}, \hat{\alpha}^*, \hat{\eta}, \hat{\eta}^*) \text{ is a saddle point of } \mathcal{L} \quad \Leftrightarrow \quad (\hat{\omega}, \hat{b}, \hat{\underline{\xi}}, \hat{\overline{\xi}},) \text{is a minimizer of 2}$$

- If $(\hat{\omega}, \hat{b}, \hat{\underline{\xi}}, \hat{\overline{\xi}}, \hat{\alpha}, \hat{\alpha}^*, \hat{\eta}, \hat{\eta}^*)$ is a saddle point of the Lagrangian then the value of the primal $\overline{\mathcal{L}}(\omega, b, \underline{\xi}, \overline{\xi},)$ and the dual $\underline{\mathcal{L}}(\alpha, \alpha^*, \eta, \eta^*)$ at that point is the same:

$$\overline{\mathcal{L}}(\hat{\omega}, \hat{b}, \hat{\underline{\xi}}, \hat{\overline{\xi}}) = \underline{\mathcal{L}}(\hat{\alpha}, \hat{\alpha}^*, \hat{\eta}, \hat{\eta}^*)$$

Since our problem has a unique minimizer, there is a saddle point and it is unique saddle point. It is thus sufficient to find that point to solve the problem. If we focus on the dual problem:

$$\underline{\mathcal{L}}(\alpha, \alpha^*, \eta, \eta^*) = \inf_{\omega \in \mathbb{R}^p, b \in \mathbb{R}} \mathcal{L}(\omega, b, \alpha, \alpha^*, \eta, \eta^*)$$

The Lagrangian of the problem is a strictly convex function, continuously differentiable on its domain

$$(\omega, b, \underline{\xi}, \overline{\xi}, \alpha, \alpha^*, \eta, \eta^*) \in \mathbb{R}^p \times \mathbb{R} \times \mathbb{R}^n_+ \times \mathbb{R}^n_+ \times \mathbb{R}^n_+ \times \mathbb{R}^n_+ \times \mathbb{R}^n_+ \times \mathbb{R}^n_+$$

From convex optimization theory, we know a necessary and sufficient condition for a point of its domain to be a minimizer based on the gradient.

$$(\omega^*, b^*, \underline{\xi}^*, \overline{\xi}^*) \in \underset{\omega \in \mathbb{R}^p, b \in \mathbb{R}, \underline{\xi} \in \mathbb{R}^n, \overline{\xi} \in \mathbb{R}^n}{\operatorname{argmin}} \mathcal{L}(\omega, b, \underline{\xi}, \overline{\xi} \alpha, \alpha^*, \eta, \eta^*) \quad \Leftrightarrow \quad \nabla \mathcal{L}(\omega^*, \beta^*, \underline{\xi}^*, \overline{\xi}^*) = 0\}$$

The left condition is satisfied for x^*, b^* solution of the following equation:

$$\nabla \mathcal{L}(\omega^*, \beta^*) = 0$$

$$\Leftrightarrow \quad \left\{ \begin{array}{rcl} \nabla_\omega \, \mathcal{L}(\omega, b, \underline{\xi}, \overline{\xi}, \alpha, \alpha^*, \eta, \eta^*) & = & 0 \\ \nabla_b \, \mathcal{L}(\omega, b, \underline{\xi}, \overline{\xi}, \alpha, \alpha^*, \eta, \eta^*) & = & 0 \\ \nabla_{\underline{\xi}} \, \mathcal{L}(\omega, b, \underline{\xi}, \overline{\xi}, \alpha, \alpha^*, \eta, \eta^*) & = & 0 \\ \nabla_{\overline{\xi}} \, \mathcal{L}(\omega, b, \underline{\xi}, \overline{\xi}, \alpha, \alpha^*, \eta, \eta^*) & = & 0 \end{array} \right. \Leftrightarrow \quad \left\{ \begin{array}{rcl} \omega + \sum_{i=1}^n (\alpha_i - \alpha_i^*) x_i & = & 0 \\ \sum_{i=1}^n (\alpha_i - \alpha_i^*) & = & 0 \\ \forall i \in [\![1, n]\!], \quad C - \alpha_i - \eta_i & = & 0 \\ \forall i \in [\![1, n]\!], \quad C - \alpha_i^* - \eta_i^* & = & 0 \end{array} \right.$$

[2] Because we minimize a proper and coercive function over a closed convex set *(Weierstrass theorem)*.
[3] Because the function to optimize is strictly convex.

We have not yet found the saddle point, but notice that from the conditions, we obtained an expression of $\hat{\omega}$ as a function of the Lagrange multipliers and the vectors x_i of our dataset.

$$\hat{\omega} = \sum_{i=1}^{n} (\alpha_i^* - \alpha_i) x_i$$

If on top of that we record the **slackness condition** (A.1) which, for the problem at end, state that at the saddle point:

- Given how our inequality constraints are set, only one of the two inequality conditions associated with α_i and α_i^* can be active *(because a point y_i cannot be on both sides of the ε-border of the prediction function)*. Hence,

$$\forall i \in [\![1, n]\!], \quad \alpha_i \alpha_i^* = 0$$

- For all vectors x_i such that y_i lies within the ε-border of the prediction function, both α_i and α_i^* are null.

Hence $\hat{\omega}$ is actually a linear combination of only some of the x_is of the dataset: the ones for which one of the inequality constraints related to the ε-border is active. These vectors are called the **support vector**.

By replacing doing replacements in the Lagragian $\mathcal{L}$ introducing the conditions obtained for the variables $\omega, b, \overline{\xi}$ and $\underline{\xi}$ to be such that we are at the minimum of the Lagrangian, we obtain the formulation of the **dual** problem:

$$\begin{aligned} \underline{\mathcal{L}}(\alpha, \alpha^*, \eta, \eta^*) = \frac{1}{2} \left\| \sum_{i=1}^{n} (\alpha_i^* - \alpha_i) x_i \right\|^2 & \qquad \left(\omega = \sum_{i=1}^{n} (\alpha_i^* - \alpha_i) x_i \right) \\ - \varepsilon \sum_{i=1}^{n} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i (\alpha_i - \alpha_i^*) & \qquad \left(\sum_{i=1}^{n} (\alpha_i^* - \alpha_i) = 0 \quad \text{and} \quad \forall i \in [\![1, n]\!], \quad C - \alpha_i^{(*)} - \eta_i^{(*)} = 0 \right) \end{aligned}$$

Finding the saddle point requires to find the maximizer of that function. Since this function is once again convex[4], we can use the results from convex optimization theory. Le value of the Lagrange multipliers α and α^* will indicate which vectors are the support vectors.
The problem to solve to get the saddle point is thus the following:

$$\begin{cases} \max\limits_{(\alpha, \alpha^*, \eta, \eta^*) \in (\mathbb{R}^n)^4} & \frac{1}{2} \left\| \sum_{i=1}^{n} (\alpha_i^* - \alpha_i) x_i \right\|^2 - \varepsilon \sum_{i=1}^{n} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i (\alpha_i - \alpha_i^*) \\ & = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\alpha_i^* - \alpha_i) \langle x_i | x_j \rangle (\alpha_j - \alpha_j^*) - \varepsilon \sum_{i=1}^{n} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i (\alpha_i - \alpha_i^*) \\ s.t. & \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) = 0 \end{cases} \tag{4}$$

1.2.3 Example

We have the soft border support vector algorithm on a dataset of points following a linear function and strictly within a border $\varepsilon = 1.5$ of that true function but with as parameter of the support vector algorithm:

- $\varepsilon = 2$ *(hence a "larger" than needed border size)*
- $C = 1$

[4] It is a quadratic function with linear equality constraints

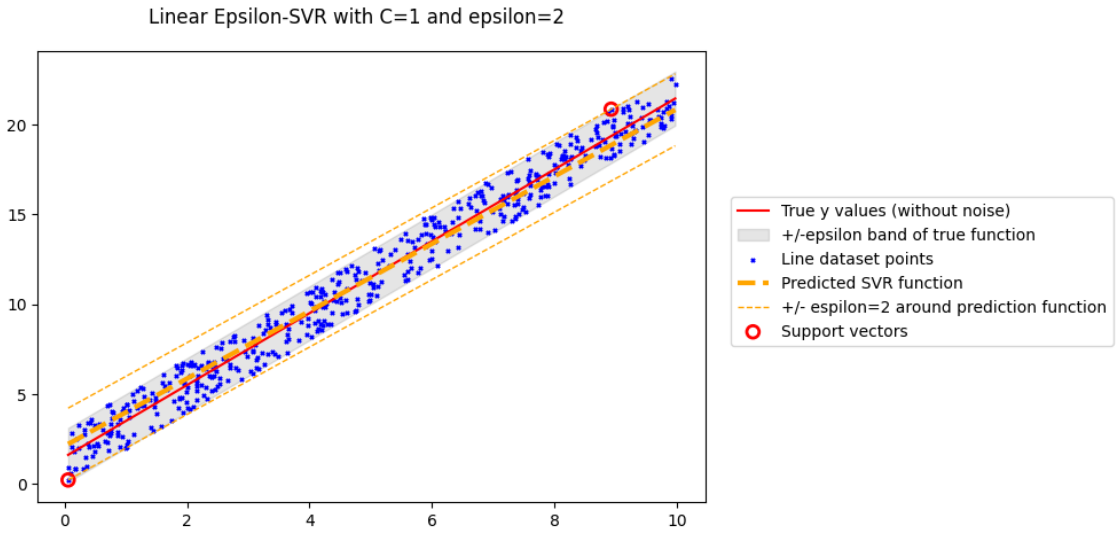


Figure 6: SVR with border taken too large

The result can be seen on Figure 6. Since the border size was chosen deliberately too large in the algorithm parameters, several linear function were such that all points are within a $\varepsilon = 2$ border of that prediction function: meaning the costs related to the distance to that prediction function are all null and thus the best function is the flattest one of all of them). One can see that the function output is the flattest one of all of these and there are only two support vectors: the two that touch the $\varepsilon = 2$ border around the flattest of all the linear functions whose border wraps the dataset.

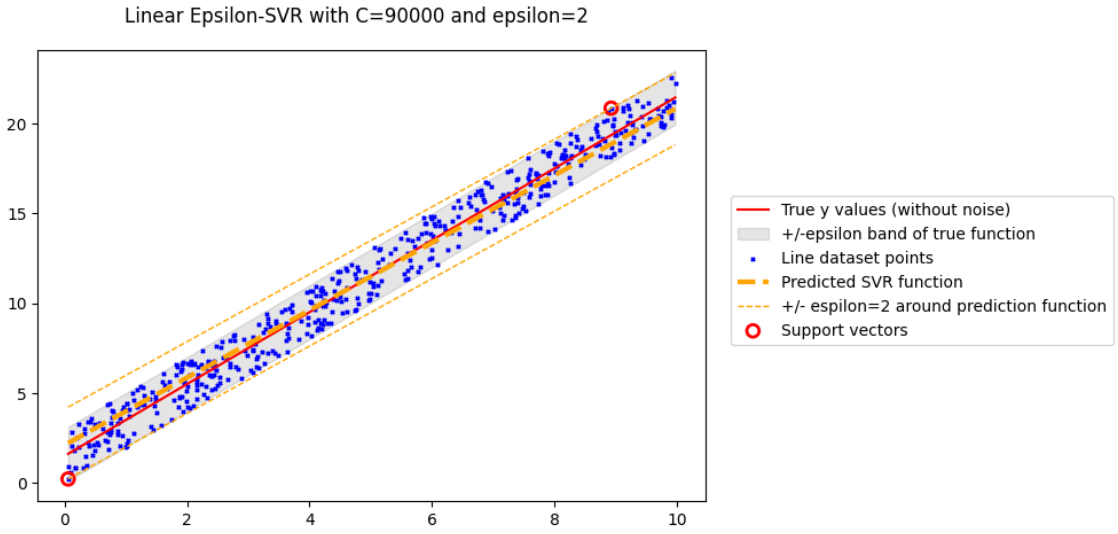


Figure 7: SVR with border taken too large but with different parameters

Also notice how the C parameter is useless if the value ε is larger than the actual spread of the datapoints around the "true" linear function. For instance, Figure 7 shows the result of running the soft border algorithm on the same dataset for $\varepsilon = 2$ again but $C = 100$. One can see that the function output is the exact same (even though the parameter C tends to favor very flat function, see again Figures 1, but here it has no effect since the associated cost can be null).

If however we reduce the ε value used in the algorithm to a value smaller than the actual border of the dataset, then there is no linear function such that all vectors are within the ε-border of the prediction function. Costs associated to the "errors" of datapoints outside the border starts to take importance.

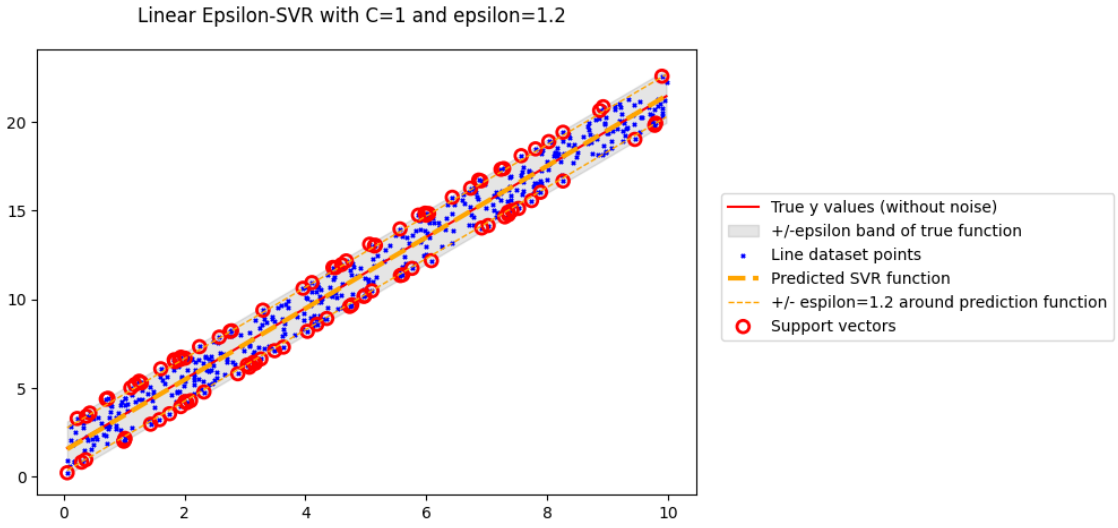


Figure 8: Soft border SVM algorithm run for ε smaller than its actual value for the dataset and $C = 1$.

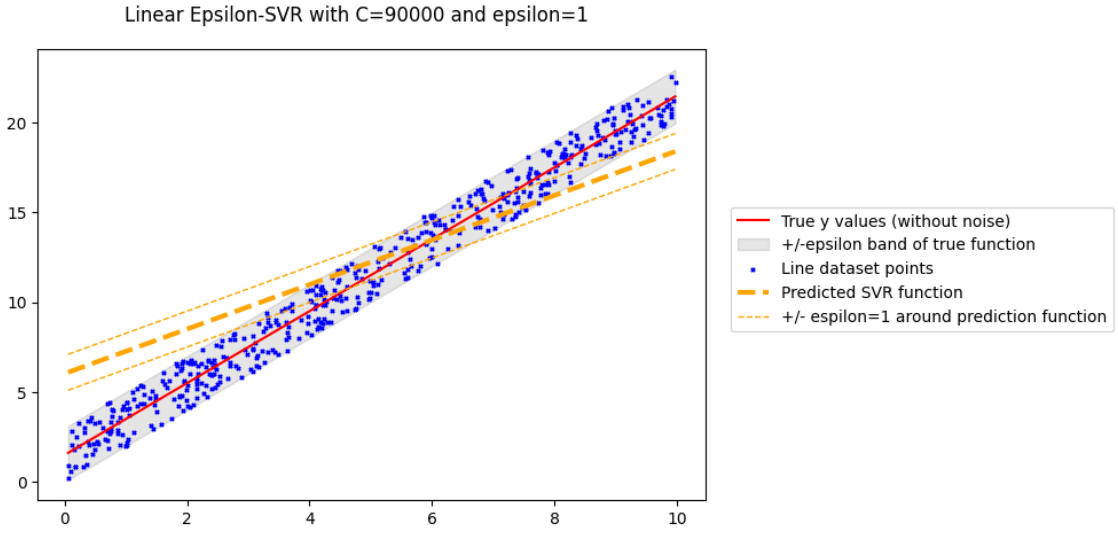


Figure 9: Soft border SVM algorithm run for ε smaller than its actual value for the dataset and C very large so that the output function starts to be very off.

2 Extension to the nonlinear setup using kernels

To solve non-linear problems without exponentially increasing the computational complexity, we utilize the **kernel trick**.

2.1 Mapping to Feature Space

The fundamental idea is to project input data from the original space $\mathcal{X}$ (e.g., 2D) into a higher-dimensional feature space $\mathcal{F}$ via a mapping $\Phi : \mathcal{X} \to \mathcal{F}$.

$$x \to \Phi(x)$$

In this high-dimensional space, data that was non-linearly distributed in $\mathcal{X}$ can often be approximated by a linear function. At this point, mapping the data into a higher-dimensional feature space is not a silver bullet because, even if we have a linear problem, it is in a much more complex space (mapping all the data can have a long computation time).

2.2 The Dual Formulation

The "kernel trick" relies on the observation that the SVR optimization problem can be rewritten in its *Dual Form* using Lagrange multipliers (α_i, α_i^*) (see Appendix A.1). Therefore, one remarks that the algorithm depends only on dot products between data points, not on the data points themselves.
The prediction function becomes an expansion of support vectors:

$$f(x) = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) \langle \Phi(x_i), \Phi(x) \rangle + b \tag{5}$$

Instead of computing the complex mapping $\Phi(x)$ explicitly, we substitute the dot product $\langle \Phi(x_i), \Phi(x) \rangle$ with a **kernel function** $k(x_i, x)$:

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle \tag{6}$$

In the primal problem, the complexity was linked to the **dimension** of the feature space, but in the dual problem, the complexity lies in the **number of support vectors**.

> **Remark :** we are now looking for the flattest function in the **feature space**, not in the input space.

2.3 Explicit Kernel Examples

By replacing the standard dot product with a kernel function, the SVR algorithm can model various non-linear structures. Below are the most common admissible kernels satisfying Mercer's condition [1].

2.3.1 Inhomogeneous polynomial Kernel

This kernel allows the modeling of interactions between features up to a specified degree p.

$$k(x, x') = (\langle x, x' \rangle + c)^p \tag{7}$$

where $p \in \mathbb{N}$ is the degree of the polynomial and $c \geq 0$ is a constant. This is useful for data that follows a polynomial trend.

2.3.2 Gaussian Kernel

The Gaussian kernel (RBF) is the most popular kernel. It corresponds to an infinite-dimensional feature space and is capable of capturing localized variations. Moreover, it is a translation-invariant kernel ($k(x, x') = k(x - x')$), which is a widespread type of kernel.

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right) \tag{8}$$

The parameter σ controls the width of the Gaussian. A small width allows the model to capture fine details (potential overfitting), while a large width smooths the function.

2.4 Architecture of a nonlinear support vector regression

The architecture of a nonlinear SVM is described in 10.

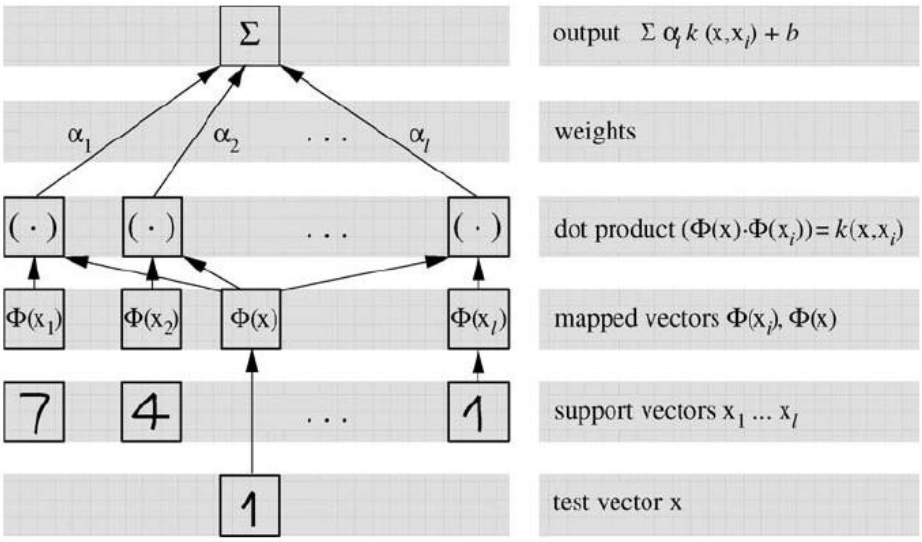


Figure 10: Architecture of a regression machine constructed by the SV algorithm [1]

The input data x is mapped into the feature space. Then the dot product is calculated with every training data so that we evaluate every $k(x_i, x)$. Finally, using the pre-trained weight α_i and the constant term b, we obtain the prediction.

2.4.1 Example

We are now going to test our method on the simple example of fitting a noisy cardinal sin (the noise is a centered Gaussian noise with $\sigma = 0.05$). To achieve that, it is possible to use a Gaussian kernel because it allows the model to fit the nonlinear variation of the sinc around each point.
For the regression, one has 3 parameters to ajust :

- C: This coefficient was defined in the first part. It quantifies the trade-off between the simplicity of the curve (smoothness) and minimizing the error. It can be compared to the hyperparameter (often noted λ) in the Ridge or Lasso regulation in linear regression
  - High C-value: The model is strict. It tries to pass through all the points, even if it creates a risk of overfitting.

  - Low C-value: The model is more tolerant of errors. It prefers a smoother, simpler curve, even if it misses some points.

- ϵ: If a point is located less than epsilon from the predicted curve, the error is considered zero.A small epsilon requires the model to be very precise (a very narrow tube). A large epsilon would allow for a coarser approximation.

- $\gamma = \frac{1}{\sigma^2}$ is specific to the Gaussian kernel:
  - High gamma: The Gaussian curve around each point is very narrow (sharp peak). The influence is very localized. The final curve will be very bumpy.
  - Low gamma: The Gaussian curve is large. The influence of a point is felt far away. The final curve will be very smooth (almost linear if gamma is close to 0).

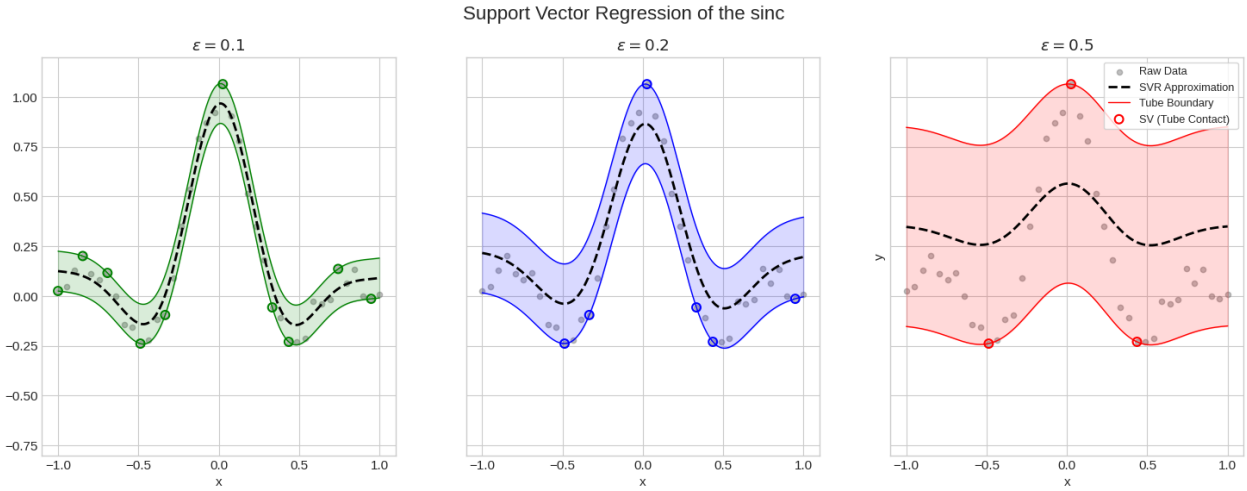


Figure 11: Regression of a sinc using SVR for different values of ϵ (C=100,$\gamma = 10$)

When the value of epsilon is changed, the width of the "tube" varies the same way. One can clearly see that the number of support vectors increases when the value of epsilon decreases because the tube is smaller and there are more vectors touching (or outside) the borders.

C is large enough so points outside the borders are penalized and so the prediction fits the curve. However, if the value of C is reduced, the influence of ϵ is less important (Figure 12):

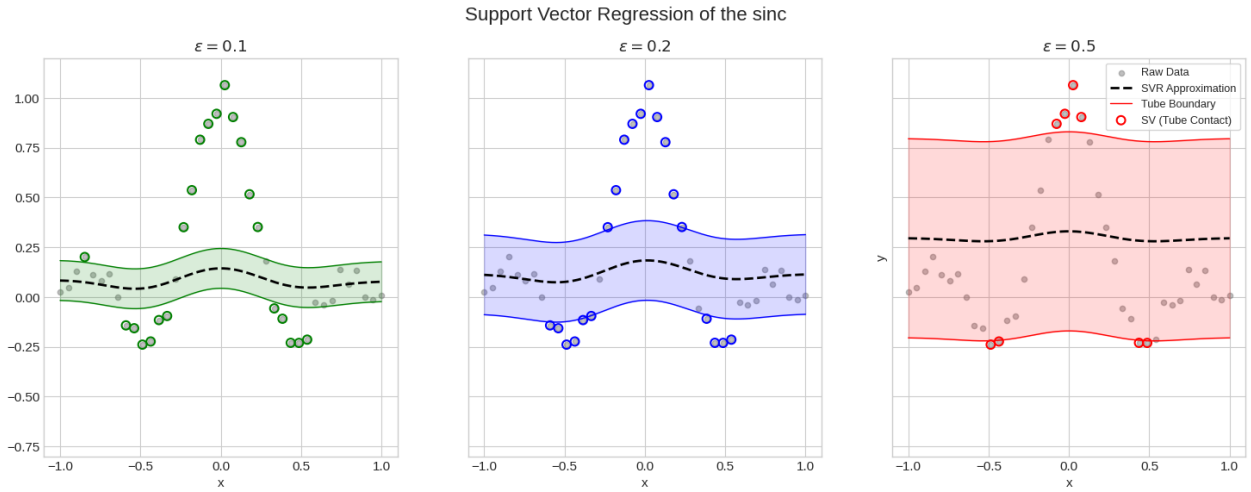


Figure 12: Regression of a sinc using SVR for different values of ϵ (C=0.01,$\gamma = 10$)

Finally, the value of γ is linked to the kernel. If γ is too small, the SVM cannot follow the high variations of the curve of the function. The result is just a large Gaussian around the points which have the bigger weight (in our case center points) (13).

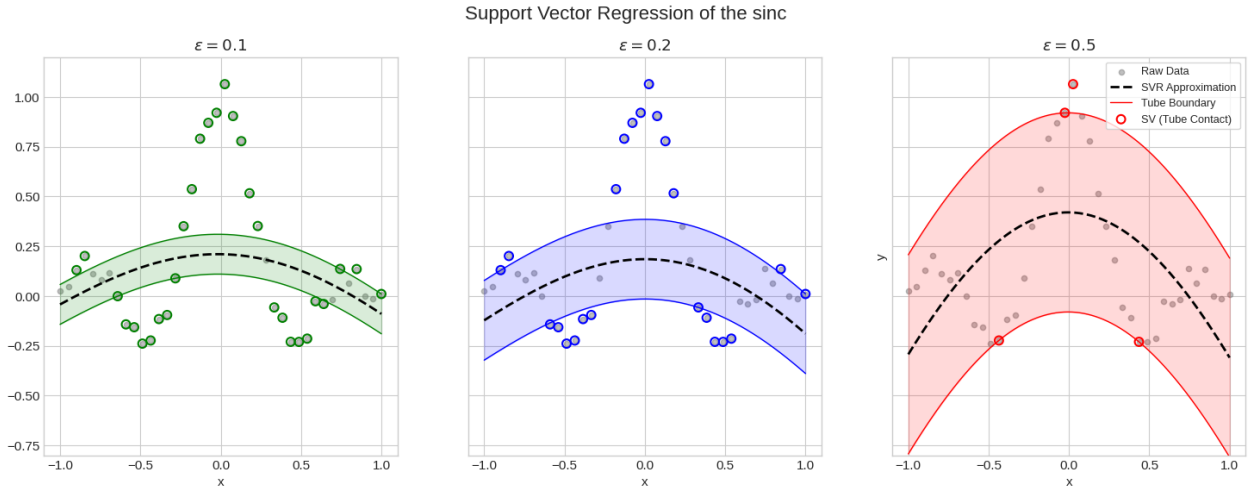


Figure 13: Regression of a sinc using SVR for different values of ϵ (C=100,$\gamma = 0.01$)

However, computing the regression with a too large γ will create unnecessary height variations around each support vector (Figure 14).

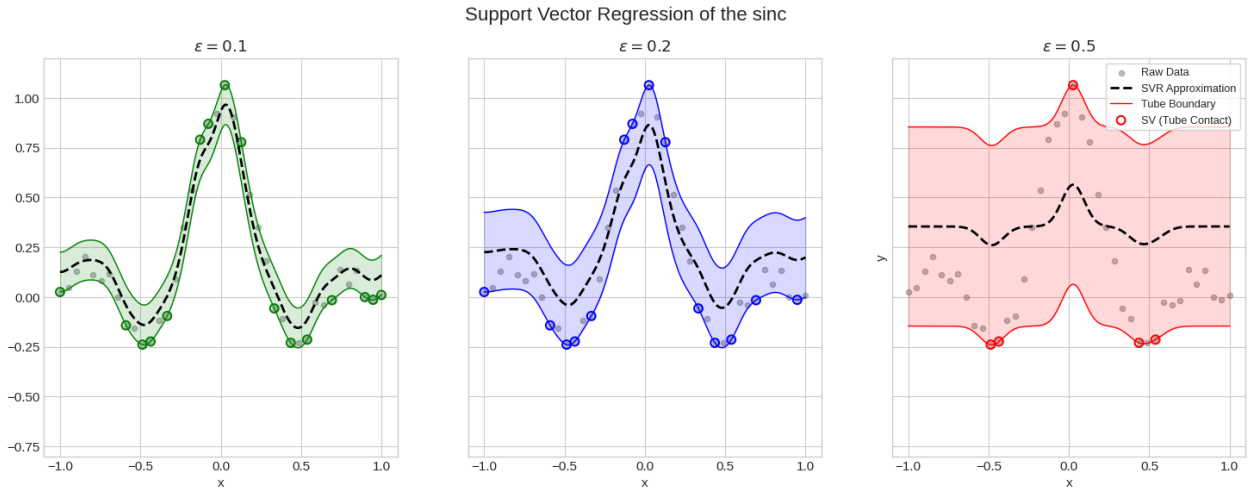


Figure 14: Regression of a sinc using SVR for different values of ϵ (C=100,$\gamma = 100$)

2.4.2 Tuning parameters and comparison with linear regression

One has to adjust every parameter seen before to have the best-fitting curve (according to the mean square error). One can compare this with the work made for a polynomial ridge regression in LAB 2. The parameters were the degree of the polynomial and the hyperparameter λ. In fact, the fundamental difference between both is that in linear regression, we compute the error for every point, so there is no ϵ to tune.
Using cross-validation, we can fine-tune our model and compare it with linear regression (Figure 15:

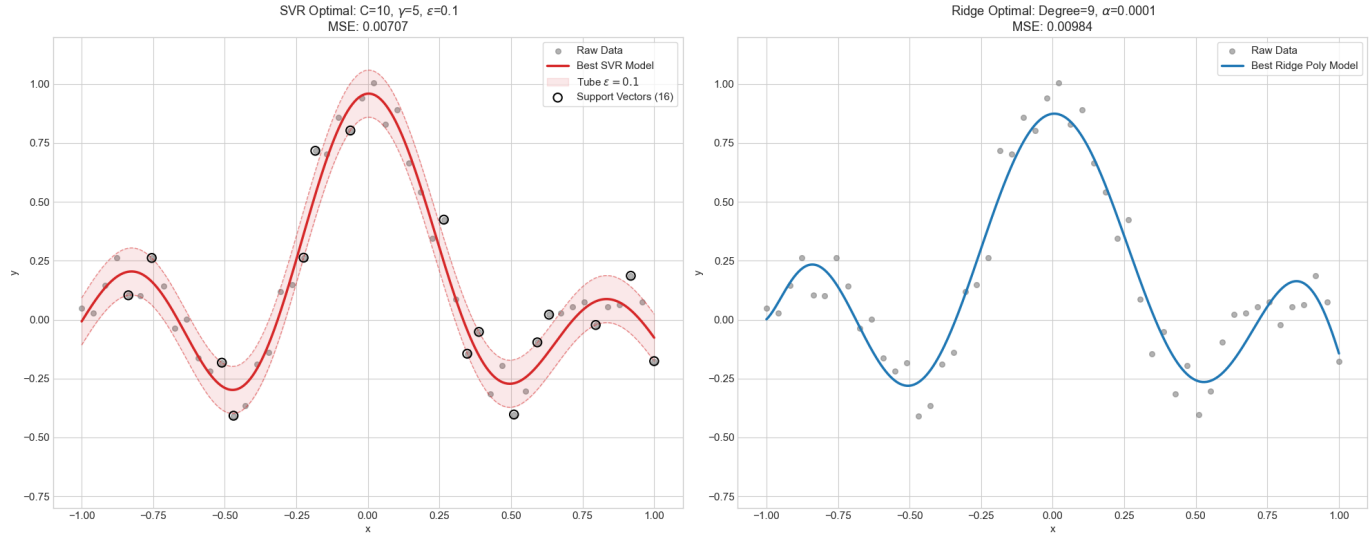


Figure 15: Comparison of SVR and linear regression with optimized parameters

The MSE of each technique is in the same order of magnitude (even if SVM is a little better, we cannot conclude anything based only on this example).

3 Cost Functions

The Support Vector (SV) algorithm for regression can be cast into standard mathematical notation, revealing connections to existing function estimation methods like Regularization Networks (explained in the next section). This framework allows for the use of statistical concepts to select suitable cost functions.

3.1 The Risk Functional

The regression problem is formulated as the minimization of a risk functional. Assuming the training data $X = \{(x_1, y_1), \ldots, (x_l, y_l)\}$ is drawn independently and identically distributed (i.i.d.) from a probability distribution $P(x, y)$, the goal is to minimize the expected risk $R[f]$:

$$R[f] = \int c(x, y, f(x))\, dP(x, y) \tag{9}$$

where $c(x, y, f(x))$ is a cost function determining the penalty for estimation errors.

$$R_{emp}[f] := \frac{1}{l} \sum_{i=1}^{l} c(x_i, y_i, f(x_i)) \tag{10}$$

Since $P(x, y)$ is unknown, the empirical risk $R_{emp}[f]$ (Equation 10) is used as an estimate. However, minimizing empirical risk alone can lead to overfitting, particularly in high-dimensional spaces (Figure 16).

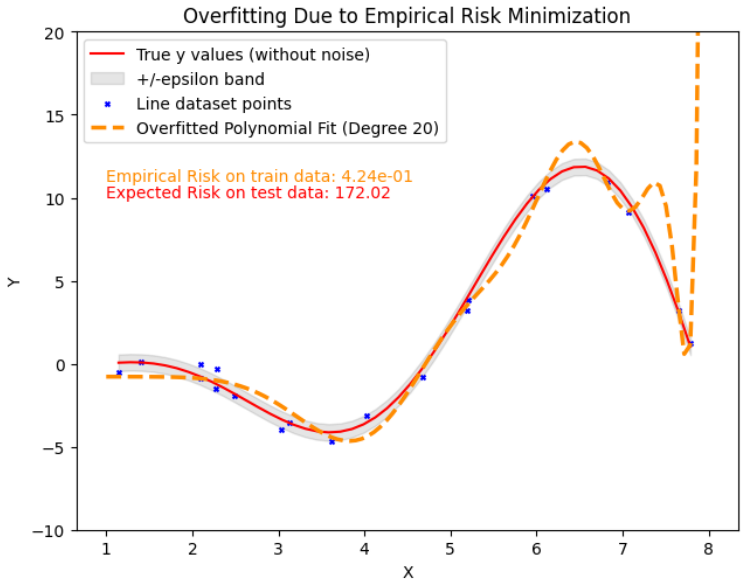


Figure 16: Resulting model when minimizing the empirical risk

To address this, a capacity control term is introduced, leading to the regularized risk functional:

$$R_{reg}[f] := R_{emp}[f] + \frac{\lambda}{2}||w||^2 \tag{11}$$

Here, λ is a regularization constant.

3.2 Maximum Likelihood and Density Models

Standard SV regression utilizes the ϵ-insensitive loss function, $c(x, y, f(x)) = |y - f(x)|_\epsilon$. Minimizing the regularized risk with this loss function is mathematically equivalent to the standard support vector optimization problem (Equation 2), provided $C = 1/(\lambda l)$.

Loss Function	$\mathbf{c}(\xi)$
ϵ-Insensitive	$\lvert\xi\rvert_\epsilon$
Laplacian (L1)	$\lvert\xi\rvert$
Gaussian (L2)	$\frac{1}{2}\xi^2$
Huber's Robust Loss	$\begin{cases} \frac{1}{2\sigma}\xi^2 & \text{if } \lvert\xi\rvert \le \sigma \\ \lvert\xi\rvert - \frac{\sigma}{2} & \text{otherwise} \end{cases}$
Polynomial	$\frac{1}{p}\lvert\xi\rvert^p$
Piecewise Polynomial	$\begin{cases} \frac{1}{p\sigma^{p-1}}\xi^p & \text{if } \lvert\xi\rvert \le \sigma \\ \lvert\xi\rvert - \sigma\frac{p-1}{p} & \text{otherwise} \end{cases}$

Table 1: Cost functions for regression

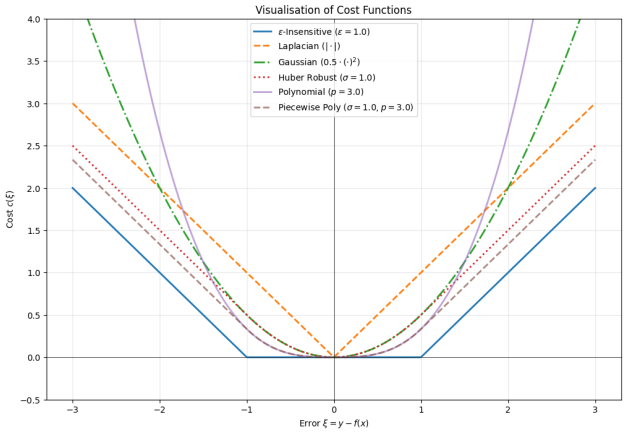


Figure 17: Visualisation of cost functions

The optimal choice of the cost function can be derived from a maximum likelihood perspective. If the data is generated by an underlying function with additive noise, $y_i = f_{true}(x_i) + \xi_i$, where the noise has a density $p(\xi)$, the optimal cost function is:

$$c(x, y, f(x)) = -\log p(y - f(x)) \tag{12}$$

3.3 Solving the Equations

The optimization problem can be generalized to a class of convex, symmetric cost functions c that are zero in the interval $[-\epsilon, \epsilon]$. By utilizing Lagrange multiplier techniques (see Appendix A.1), a generalized dual optimization problem is derived. The resulting dual maximization problem takes the form:

$$\begin{aligned} \text{maximize } & -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle \\ & + \sum_{i=1}^{l} y_i(\alpha_i - \alpha_i^*) - \epsilon(\alpha_i + \alpha_i^*) + C\sum_{i=1}^{l}(T(\xi_i) + T(\xi_i^*)) \end{aligned} \tag{13}$$

where $T(\xi)$ depends on the derivative of the cost function: $T(\xi) := \tilde{c}(\xi) - \xi\partial_\xi\tilde{c}(\xi)$.

3.4 Application

The choice of cost function isn't just a theoretical detail, it has real practical implications for how Support Vector Regression performs on actual data. Effectively, picking the right loss function comes down to finding the best compromise between statistical accuracy and computational speed.

3.4.1 Adapting to Noise:

Ideally, the cost function should reflect the nature of the noise in your data. If we are dealing with standard Gaussian noise, the classic Gaussian (L_2) loss is most suited, as it corresponds to the Maximum Likelihood Estimator.

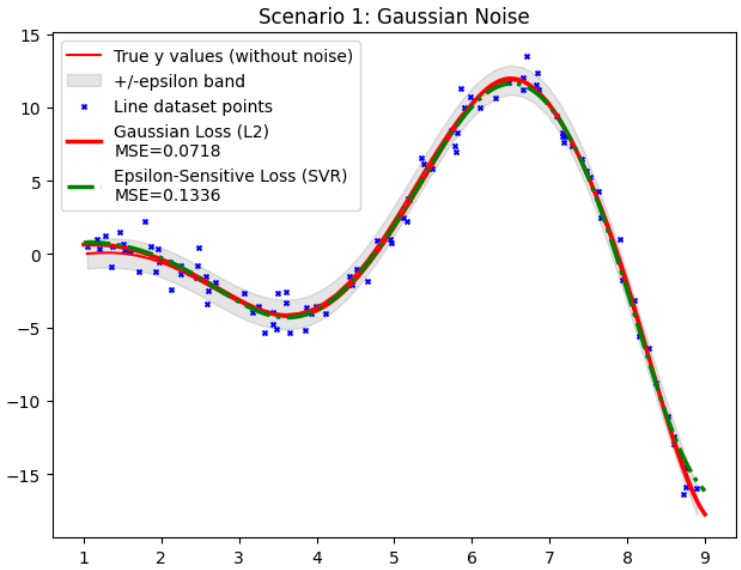


Figure 18: Gaussian Loss efficiently tracks the mean, whereas the ϵ-Insensitive Loss results in slightly higher error.

In Figure 18, we can see that the model using Gaussian loss (red) follows the true function closely. The ϵ-insensitive loss (green) does a decent job, but because it ignores data with small errors within the ϵ-tube, it discards some useful information, resulting in a higher error.

3.4.2 Handling Outliers:

When our data contains heavy outliers, we need a loss function that doesn't overreact. A loss with a bounded derivative, like the ϵ-insensitive loss, is crucial here. The Gaussian loss, by contrast, squares the errors, which gives extreme outliers a disproportionate amount of influence.

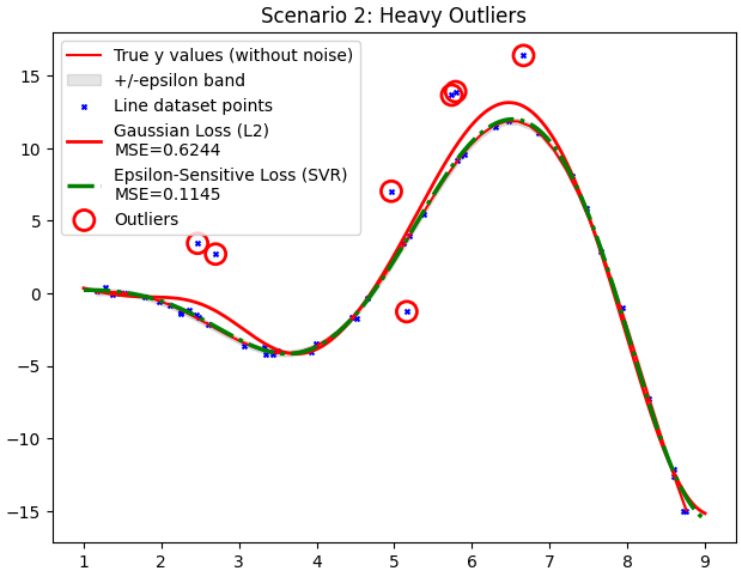


Figure 19: The ϵ-Insensitive Loss stays robust in the presence of outliers, while the Gaussian Loss deviates significantly.

Figure 19 shows why this matters. The Gaussian model (red) gets pulled heavily towards the outliers to minimize the squared error, distorting the fit. The SVR model (green) stays robust. Since its cost function only grows linearly, the "pull" of these outliers is capped (the dual variables α hit a ceiling at C), effectively allowing the model to ignore them and capture the true underlying trend.

3.4.3 The Cost of Prediction (Sparsity):

There is also an important trade-off between the choice of ϵ and prediction speed. Using $\epsilon \neq 0$ keeps the solution "sparse," meaning the model is defined by only a small subset of the training data (the Support Vectors). If we use a cost function where $\epsilon = 0$ (like the standard Gaussian loss), we lose this benefit; every single training point becomes a Support Vector. For large datasets, this makes generating new predictions incredibly slow.

4 SVMs : a usefull case of Regularization Networks

4.1 Context

Support Vector Machines (SVMs) can be viewed as a special case of Regularization Networks (RNs). As explained previously, the SVM's regularization is applied in a high-dimensional feature space using a kernel function. One shows that this kernel function corresponds to the **Green's function** of a specific **regularization operator** in the **input space**. Therefore, SVMs optimize a regularized risk, just like RNs, but with the added advantages of speed through the sparsity (due to support vectors) and the ability to handle non-linear relationships efficiently through the kernel trick. Besides, SVMs provides a more flexible solution than RNs as on can choose to customize and fine-tune kernels to suit to specific problems, leveraging smoothing properties or prior knowledge.

4.2 Principles of Regularization Networks

RNs minimize a **regularized risk** of the form:

$$R_{\text{reg}}[f] = R_{\text{emp}}[f] + \frac{\lambda}{2}\|Pf\|^2,$$

where:

- $R_{\text{emp}}[f]$ is the empirical risk.
- P is a **regularization operator** in the sense of Tikhonov and Arsenin (1977) (i.e. it is a positive semidefinite operator mapping from the Hilbert space H of functions f to a dot product space D).
- $\|Pf\|^2$ measures the "complexity" of f in the **input space**.

Unlike SVMs, which enforce *flatness* in the feature space, RNs penalize variations of f directly in the input space focusing on its *smoothness*. (e.g. tackling the well known over-fitting effect by choosing an operator that penalizes large variations of f).

4.3 Connection with SVMs: Green's Functions and Kernels

The key link between SVMs and RNs lies in the **Green's functions** G of the operator P^*P:

$$(P^*PG_{x_i})(x) = \delta_{x_i}(x),$$

where δ_{x_i} is the Dirac delta function. Green's functions G are **Mercer kernels**, and they satisfy:

$$k(x_i, x_j) = \langle (PG)(x_i, \cdot), (PG)(x_j, \cdot) \rangle.$$

Thus, a SVM kernel k can be interpreted as the Green's function of a regularization operator P. For example:

- The **Gaussian RBF kernel** $k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$ corresponds to an operator P that penalizes derivatives of f (strong smoothing).
- **B-spline kernels** (compact support) are associated with local regularization operators.

From now, one should understand that given a regularization operator P, he can find a kernel k such that its use in a SVM will not only ensure flatness in the feature space but also ensure the minimization of a P regularized risk.

4.4 Advantages of SVM

As SVMs are a special case of RNs it benefit from its theoretical converging properties and warranties. But, RNs suffer from **long computing time** (sparsity is not guaranteed) and sometimes **over-fitting** when applied to the resolution of high dimensional problems. Therefore, SVMs are a usefull case of RNs as it come with the following benefits :

- **Flexibility**: Through kernels functions SVRs handle non-linear problems
- **Adaptability (Kernel Selection)**: The connection with RNs allows choosing kernels tailored to the data structure and adapted to the smoothing requirement of the function.
- **Computational Efficiency**: For compactly supported kernels, the Gram matrix $k(x_i, x_j)$ becomes sparse, reducing complexity and computation time.

4.5 Application

The goal of this application is to compare the ability of two kernel based model to estimate the right fit to a non linear function, based on the dataset size.

The function *function a* is implemented below for three different dataset size, using parameter $\epsilon = 1.5$.

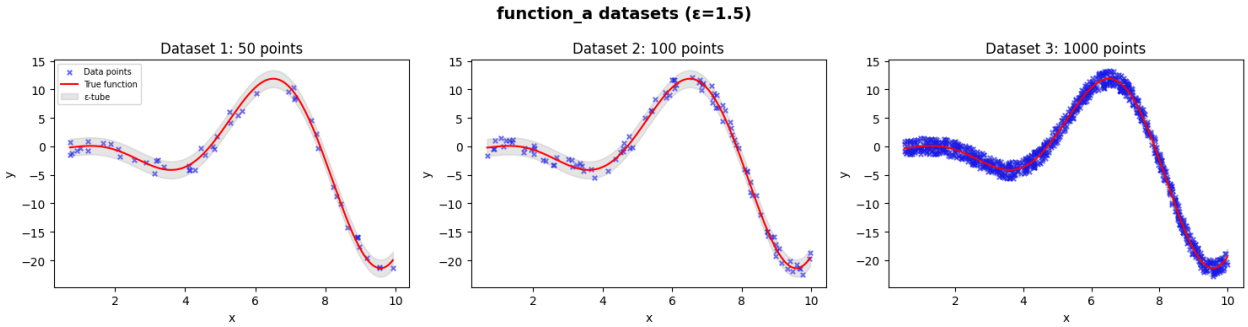


Figure 20: Three dataset : *function a* implemented with 50, 100 and 1000 points

The estimation of this function will be done by training two models based on a **RBF kernel** (of parameter $\gamma = 0.5$) :

- **ϵ-soft SVR** :
  - ϵ insensitive loss
  - minimize Equation 2:

$$\underset{\omega \in \mathbb{R}^p,\, b \in \mathbb{R}}{\operatorname{argmin}} \frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^{n} (\xi_i + \overline{\xi}_i)$$

  - quadratic optimization
- **Regularization Network** (Kernel Ridge Regression) :
  - quadratic loss
  - minimize

$$\min_{\mathbf{c}} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \|\mathbf{c}\|_K^2$$

– Linear optimization : $f(x) = \sum_{j=1}^{n} c_j K(x, x_j)$ with K being the kernel function.
The solution is obtained by solving the linear system:

$$(K + \lambda I)\mathbf{c} = \mathbf{y}$$

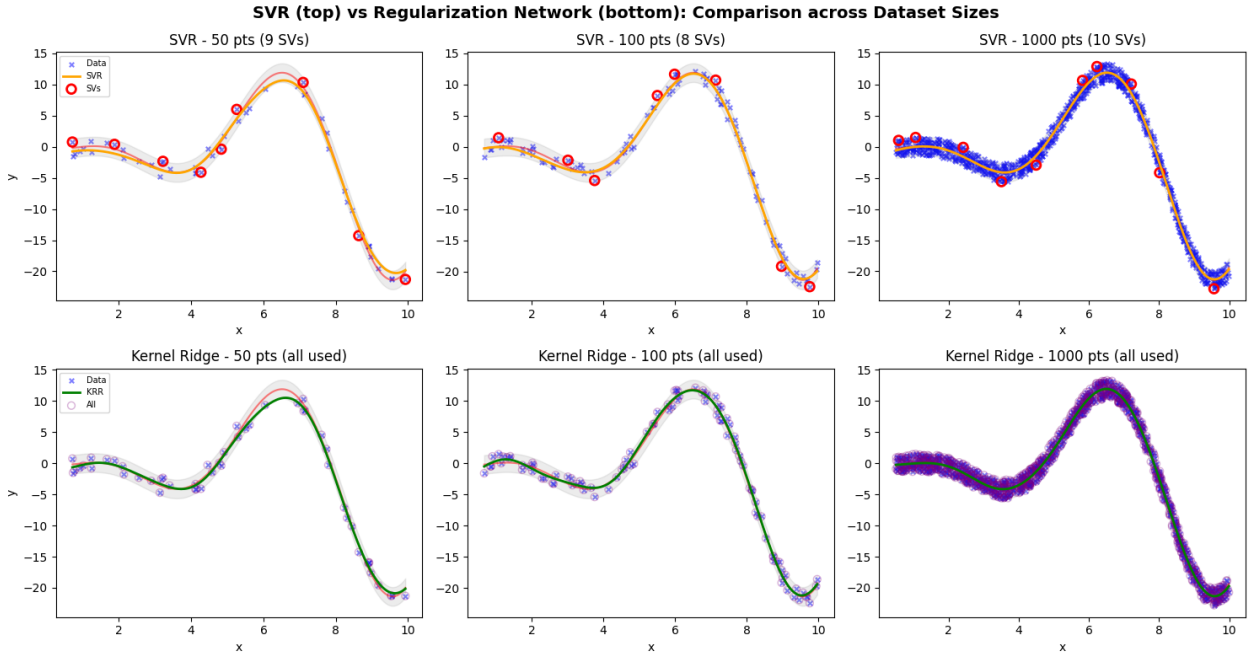


Figure 21: Plots of the results of the regressions : highlighting the points used to train the models and the resulting predicted function versus the true one

```
>>> Dataset: 50 points
------------------------------------------------------------------------
Metric                    SVR                  Kernel Ridge
------------------------------------------------------------------------
Training time (ms)        0.777                1.056
Points used               9                    50
Sparsity                  82.0%                0.0%
MSE (vs true)             0.342670             0.192140
MAE (vs true)             0.468413             0.356401

>>> Dataset: 100 points
------------------------------------------------------------------------
Metric                    SVR                  Kernel Ridge
------------------------------------------------------------------------
Training time (ms)        0.465                1.417
Points used               8                    100
Sparsity                  92.0%                0.0%
MSE (vs true)             0.073879             0.069509
MAE (vs true)             0.185862             0.220884

>>> Dataset: 1000 points
------------------------------------------------------------------------
Metric                    SVR                  Kernel Ridge
------------------------------------------------------------------------
Training time (ms)        4.029                48.127
Points used               10                   1000
Sparsity                  99.0%                0.0%
MSE (vs true)             0.002023             0.004296
MAE (vs true)             0.024807             0.056052
```

Figure 22: Computational time and model estimation quality verification

The training phase was timed for each pair model/dataset and the results are what was expected : **SVR is faster for each dataset**

But, it is also relevant to compare the MSE and MAE. Indeed, the SVR is less reliable on small dataset (regarding the chosen parameters of the SVR) than the Kernel Ridge Regression. Whereas, **for large dataset**, as it was shown in the article, **SVR performs better** than the Kernel Ridge Regression : it is **10 times faster and 2 times more accurate**.

One must remain careful as the chosen C in SVR is large. Therefore the model is made to prioritize fitting over regularization. In order to verify that it is not over fitting, one may add outliers to show that the model is robust. Indeed, the chosen RN is more sensitive to outliers than the SVR.

Conclusion

In this report, we have explored the theoretical foundations and practical applications of Support Vector Machines for regression. We began by formulating the linear problem, demonstrating the necessity of the "soft border" approach (using slack variables) to handle noise and ensure the feasibility of the optimization problem.

We extended the method to the nonlinear setup using the kernel trick, which allows for efficient mapping into high-dimensional feature spaces without explicit computation. Our analysis of cost functions highlighted the trade-off between statistical efficiency and robustness. Specifically, the ϵ-insensitive loss function proved superior in handling outliers compared to the standard Gaussian (L_2) loss, primarily due to the sparsity of the resulting model.

Finally, we established the connection between SVMs and Regularization Networks. Through numerical experiments, we confirmed that while both methods provide comparable accuracy, SVR benefits significantly from sparsity. This results in superior computational efficiency on larger datasets compared to Kernel Ridge Regression, making SVR a robust and flexible tool for function estimation.

A Appendix

A.1 Lagrange multiplier methods

As a reminder:

Definition A.1: Primal, dual and saddle point of the Lagrangian

For an optimization problem of the form

$$\begin{cases} \min_{x\in\mathbb{R}^n} & f(x) \\ s.t. & \begin{cases} \forall i \in [\![1,m]\!] & g_i(x) = 0 \\ \forall j \in [\![1,p]\!] & h_j(x) \leq 0 \end{cases} \end{cases}$$

The associated **Lagrangian** is the function

$$\mathcal{L} : \begin{cases} \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}_+^p & \to & \mathbb{R} \\ (x,\lambda,\mu) & \mapsto & f(x) + \sum_{i=1}^m \lambda_i\, g_i(x) + \sum_{j=1}^p \mu_j\, h_j(x) \end{cases}$$

The **primal** problem is

$$\overline{\mathcal{L}}(x) = \sup_{\lambda\in\mathbb{R}^m, \mu\in\mathbb{R}_+^p} \mathcal{L}(x,\lambda,\mu)$$

The **dual** problem is

$$\underline{\mathcal{L}}(x) = \inf_{x\in\mathbb{R}^p} \mathcal{L}(x,\lambda,\mu)$$

A point $(\hat{\omega}, \hat{b}, \hat{\alpha}, \hat{\alpha}^*, \hat{\eta}, \hat{\eta}^*)$ is a **saddle point** of the Lagrangian if and only if:

- **It is a minimizer of the primal problem:**

$$\forall x \in \mathbb{R}^n, \quad \overline{\mathcal{L}}(\hat{x}) \leq \overline{\mathcal{L}}(x)$$

- **It is a maximizer of the dual problem:**

$$\forall (\lambda,\mu) \in \mathbb{R}^m \times \mathbb{R}_+^p, \quad \underline{\mathcal{L}}(\lambda,\mu) \leq \underline{\mathcal{L}}(\hat{\lambda},\hat{\mu})$$

- At this point the value of the primal and the dual are equal:

$$\underline{\mathcal{L}}(\hat{\lambda},\hat{\mu}) = \overline{\mathcal{L}}(\hat{x})$$

- **The complementary slackness holds:**

$$\forall j \in [\![1,p]\!], \quad \mu_j\, h_j(x) = 0$$

  Meaning that at the saddle point, for each of the inequality constraints:

  - Either the inequality constraint is "**active**", that is we are at the "limit" for the feasible set of points for that inequality constraint:

$$h_j(x) = 0$$

  - Or (if $h_j(x) < 0$), the Lagrange multipliers μ_j associated to the constraint is null:

$$\mu_j = 0$$

References

[1] Alex J. Smola and Bernhard Schölkopf. "A tutorial on support vector regression". In: *Statistics and Computing* 14.3 (2004), pp. 199–222. DOI: https://doi.org/10.1023/B:STCO.0000035301.49549.88.