# Méthodes SVM pour l'identification

Fabien Lauer, Gérard Bloch

# Méthodes SVM pour l'identification

Fabien Lauer, Gérard Bloch

Centre de Recherche en Automatique de Nancy (CRAN UMR 7039), Nancy Université, CNRS
CRAN-ESSTIN, rue Jean Lamour, 54519 Vandœuvre Cedex

fabien.lauer@esstin.uhp-nancy.fr, gerard.bloch@esstin.uhp-nancy.fr
http://www.cran.uhp-nancy.fr

**Résumé**— *Ce papier présente les méthodes SVM (Support Vector Machines), issues de la théorie de l'apprentissage statistique, pour l'identification des systèmes. La méthode est tout d'abord exposée dans le cas linéaire, puis étendue au cas non-linéaire grâce à des fonctions noyaux. Ces noyaux permettent de résoudre le problème non-linéaire par une projection implicite dans un espace de dimension supérieure. Le papier traite ensuite des questions pratiques telles que le réglage des hyperparamètres ou l'implémentation algorithmique de la méthode. Les liens avec les réseaux de neurones sont explicités de manière à mettre en valeur les avantages de la méthode : sélection automatique des centres, régularisation intrinsèque, unicité de la solution et bonne généralisation à partir de peu de données. La méthode est illustrée sur un exemple d'identification de système non-linéaire et comparée à un des algorithmes les plus performants, basé sur la sélection des centres des réseaux RBF par Orthogonal Least Squares (OLS) et l'estimation de l'erreur de généralisation.*

**Mots-clés**— *identification, machines d'apprentissage, SVM, fonctions noyaux, réseaux de neurones.*

## I. Introduction

Le but de ce papier est de présenter les méthodes dites SVM (*Support Vector Machine*), issues de la théorie de l'apprentissage statistique [41], pour l'identification des systèmes. La théorie de l'apprentissage sur laquelle s'appuient les SVMs permet de définir des bornes pour l'erreur du modèle sur des données futures, i.e. non disponibles lors de l'apprentissage (erreur de généralisation). La minimisation de ces bornes conduit à l'algorithme d'apprentissage SVM qui ne minimise donc pas simplement l'erreur sur les données disponibles comme la plupart des algorithmes d'apprentissage de réseaux de neurones. Les SVMs, ou encore Machines à Vecteurs de Support, ont originellement été développées pour la reconnaissance de formes ou la classification. Dans ces domaines, elles se sont rapidement imposées comme l'état de l'art tant sur des benchmarks que sur diverses applications [32], [34], [42], [19], [23], [24]. Une formulation des SVMs pour la régression, appelée SVR (*Support Vector Regression*), existe aussi. Cette méthode a récemment été appliquée avec succès sur un certain nombre de problèmes d'identification [14], [5], [26], [45], [1], [43]. Certains auteurs ont aussi proposé des améliorations de la méthode pour son application spécifique à l'identification [10], [18].

Ce papier présente la méthode SVM pour la régression tout d'abord pour le cas linéaire (Section II-A), puis pour le cas non-linéaire (Section II-B) dans sa forme originelle qui conduit à un problème de programmation quadratique. Une autre formulation conduisant à un problème de programmation linéaire est aussi exposée (Section II-C). La Section suivante traite des questions liées à l'application pratique de la méthode comme le réglage des hyperparamètres (Sect. III-A et III-B), l'implémentation algorithmique (Sect. III-C) ou l'application concrète à l'identification des systèmes (Sect. III-D). La Section IV met en valeur les avantages des SVMs par rapport aux approches plus classiques pour les réseaux de neurones avant de proposer une éventuelle extension pour la prise en compte de connaissances a priori. Ces avantages sont évalués en Section V sur un exemple d'identification de système non-linéaire.

*Notations :* Les caractères gras représentent des vecteurs ou des matrices. Tous les vecteurs sont des vecteurs colonne et sont notés en lettres minuscules. Les matrices sont représentées par des lettres majuscules. Pour $\boldsymbol{A} \in \mathbb{R}^{p\times m}$ et $\boldsymbol{B} \in \mathbb{R}^{p\times n}$ contenant des vecteurs de dimension $p$, le "noyau" $\boldsymbol{K}(\boldsymbol{A},\boldsymbol{B})$ projette $\mathbb{R}^{p\times m}\times\mathbb{R}^{p\times n}$ dans $\mathbb{R}^{m\times n}$ avec $\boldsymbol{K}(\boldsymbol{A},\boldsymbol{B})_{i,j} = k(\boldsymbol{A}_i,\boldsymbol{B}_j)$, où $k : \mathbb{R}^{p\times p} \to \mathbb{R}$ est la fonction noyau. En particulier, si $\boldsymbol{x} \in \mathbb{R}^p$ est un vecteur colonne alors $\boldsymbol{K}(\boldsymbol{x},\boldsymbol{B})$ est un vecteur ligne dans $\mathbb{R}^{1\times n}$. La matrice $\boldsymbol{X} \in \mathbb{R}^{N\times p}$ contient tous les exemples d'apprentissage $\boldsymbol{x}_i$, $i = 1,\ldots,N$, en tant que lignes. Le vecteur $\boldsymbol{y} \in \mathbb{R}^N$ rassemble toutes les valeurs désirées $y_i$ pour ces exemples. La matrice de noyau $\boldsymbol{K}(\boldsymbol{X}^T,\boldsymbol{X}^T)$ est notée simplement $\boldsymbol{K}$. Le produit scalaire est noté $\langle .,. \rangle$ .

## II. Régression par Machines à Vecteurs de Support (SVR)

La régression par Machines à Vecteurs de Support (SVR) [41] consiste à trouver la fonction $f(x)$ qui a au plus une déviation $\varepsilon$ par rapport aux exemples d'apprentissage $(\boldsymbol{x}_i, y_i)$, pour $i = 1,\ldots,N$ , et qui est le plus plate possible. Cela revient à ne pas considérer les erreurs inférieures à $\varepsilon$ et à interdire celles supérieures à $\varepsilon$ [40]. Maximiser la platitude de la fonction permet de minimiser la complexité du modèle qui influe sur ses performances en généralisation. En effet, la théorie de l'apprentissage [41] permet de borner l'erreur de généralisation par une somme de deux termes : l'un dépendant de la complexité du modèle et l'autre dépendant de l'erreur sur les données d'apprentissage [11]. Les méthodes SVMs sont basées sur le contrôle de la complexité du modèle lors de l'apprentissage. Les Sections suivantes présentent l'algorithme SVR pour un modèle linéaire, puis pour le cas non-linéaire.

*A. Problème linéaire*

La méthode est d'abord décrite pour une fonction $f$ linéaire de la forme

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b \ , \tag{1}$$

avec $\boldsymbol{x} \in \mathbb{R}^p$ le vecteur d'entrée, $\boldsymbol{w} \in \mathbb{R}^p$ le vecteur des paramètres (ou poids) et $b$ une constante à déterminer. Pour assurer la platitude de la fonction $f$, la norme des poids $\|\boldsymbol{w}\|$ est minimisée (contrainte sur la dérivée de la fonction $f$). Le problème revient donc à minimiser cette norme en garantissant que les erreurs sont inférieures à $\varepsilon$ et peut s'écrire

$$\begin{aligned} \min \quad & \tfrac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{s.c.} \quad & |y_i - \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle - b| \leq \varepsilon, \ i = 1, \ldots, N \ . \end{aligned} \tag{2}$$

On remarque que dans cette formulation du problème, l'objectif n'est pas de minimiser l'erreur d'apprentissage comme dans les réseaux de neurones ou la plupart des algorithmes de régression, mais d'assurer la platitude de la fonction. La minimisation de l'erreur n'apparaît que sous forme de contraintes, ici inviolables. Autrement dit, aucune erreur n'est autorisée. Cette description du problème considère donc qu'une fonction linéaire $f$ qui approxime tous les exemples avec une précision $\varepsilon$ existe. Dans la pratique, ce n'est pas toujours le cas. En présence de bruit trop important ou de valeurs aberrantes, il est aussi plus important d'autoriser certaines erreurs. Dans ce cas, le concept de marge souple (*soft margin*) est utilisé. Il consiste à introduire des variables de relâchement (*slack variables*) $\xi_i$, $\xi_i^*$ pour rendre faisables les contraintes du problème d'optimisation qui devient

$$\min \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{i=1}^{N} (\xi_i + \xi_i^*) \ , \tag{3}$$

sous les contraintes

$$\begin{aligned} y_i - \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle - b &\leq \varepsilon + \xi_i \\ \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b - y_i &\leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* &\geq 0 \ , \ i = 1, \ldots, N \ , \end{aligned} \tag{4}$$

où $\xi_i$ et $\xi_i^*$ représentent respectivement les erreurs positives et négatives. La constante $C > 0$ est un hyperparamètre permettant de régler le compromis entre la quantité d'erreur autorisée et la platitude de la fonction $f$. Cette formulation du problème revient à utiliser une fonction d'erreur $|\xi|_\varepsilon$ appelée $\varepsilon$-insensible ($\varepsilon$-*insensitive*) de la forme

$$|y - f(\boldsymbol{x})|_\varepsilon = \begin{cases} 0 & , \text{pour } |y - f(\boldsymbol{x})| \leq \varepsilon \\ |y - f(\boldsymbol{x})| - \varepsilon & , \text{pour } |y - f(\boldsymbol{x})| > \varepsilon \ , \end{cases} \tag{5}$$

et représentée Figure 1. On peut interpréter cette fonction comme créant un tube d'insensibilité de rayon $\varepsilon$ autour de la fonction $f(\boldsymbol{x})$ (voir la Figure 2). Les variables $\xi_i$ ou $\xi_i^*$ représentent alors la distance selon l'axe $y$ entre le point $(\boldsymbol{x}_i, y_i)$ et le bord ce tube.

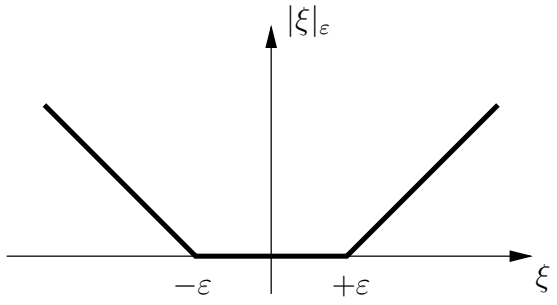


Fig. 1. La fonction d'erreur $\varepsilon$-*insensible*.

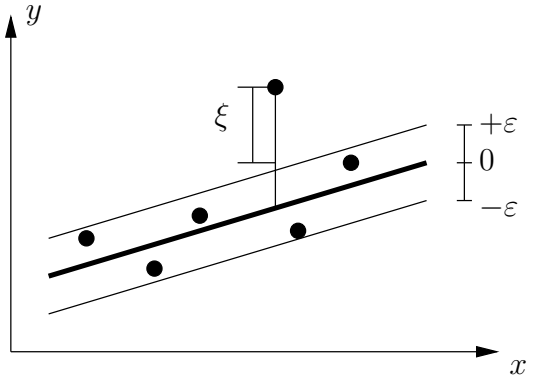


Fig. 2. Tube d'insensibilité de rayon $\varepsilon$.

Le problème (3) se résout en minimisant le Lagrangien $L$ donné par

$$\begin{aligned} L = & \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{i=1}^{N} (\xi_i + \xi_i^*) - \sum_{i=1}^{N} (\eta_i \xi_i + \eta_i^* \xi_i^*) \\ & - \sum_{i=1}^{N} \alpha_i (\varepsilon + \xi_i - y_i + \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \\ & - \sum_{i=1}^{N} \alpha_i^* (\varepsilon + \xi_i^* + y_i - \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle - b) \ , \end{aligned} \tag{6}$$

où les $\eta_i$, $\eta_i^*$, $\alpha_i$, $\alpha_i^*$ positifs représentent les multiplicateurs de Lagrange. Le calcul des dérivées de $L$ par rapport aux variables primales $(\boldsymbol{w}, b, \xi_i, \xi_i^*)$ donne

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{N} (\alpha_i^* - \alpha_i) = 0 \tag{7}$$

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{N} (\alpha_i^* - \alpha_i) \boldsymbol{x}_i = 0 \tag{8}$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \eta_i = 0 \tag{9}$$

$$\frac{\partial L}{\partial \xi_i^*} = C - \alpha_i^* - \eta_i^* = 0 \ . \tag{10}$$

Les équations (9) et (10) permettent de supprimer les variables $\eta_i = C - \alpha_i$ et $\eta_i^* = C - \alpha_i^*$. En réinjectant ces résultats dans (6), on obtient le Lagrangien dual qui doit

être maximisé :

$$L_D = -\frac{1}{2}\sum_{i,j=1}^{N}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\langle \boldsymbol{x}_i,\boldsymbol{x}_j\rangle - \varepsilon\sum_{i=1}^{N}(\alpha_i+\alpha_i^*)+\sum_{i=1}^{N}y_i(\alpha_i-\alpha_i^*)\ , \tag{11}$$

sous les contraintes

$$\sum_{i=1}^{N}(\alpha_i-\alpha_i^*)=0 \quad \text{et} \quad \alpha_i,\alpha_i^*\in[0,C]\ . \tag{12}$$

A partir de (8), les poids du modèle sont déterminés par

$$\boldsymbol{w}=\sum_{i=1}^{N}(\alpha_i-\alpha_i^*)\boldsymbol{x}_i\ , \tag{13}$$

et le modèle s'écrit donc

$$f(\boldsymbol{x})=\sum_{i=1}^{N}(\alpha_i-\alpha_i^*)\langle \boldsymbol{x}_i,\boldsymbol{x}\rangle+b\ . \tag{14}$$

Le paramètre de biais $b$ peut se calculer grâce aux conditions de Karush-Kuhn-Tucker (KKT) selon lesquelles les produits entre les variables duales et les contraintes sont nuls à la solution :

$$\alpha_i(\varepsilon+\xi_i-y_i+\langle \boldsymbol{w},\boldsymbol{x}_i\rangle+b)=0 \tag{15}$$
$$\alpha_i^*(\varepsilon+\xi_i^*+y_i-\langle \boldsymbol{w},\boldsymbol{x}_i\rangle-b)=0 \tag{16}$$
$$\eta_i\xi_i=(C-\alpha_i)\xi_i=0 \tag{17}$$
$$\eta_i^*\xi_i^*=(C-\alpha_i^*)\xi_i^*=0\ . \tag{18}$$

$b$ peut par exemple être déterminé par (15) sur un point particulier pour lequel $\boldsymbol{x}_i,\ y_i$ et $\xi_i$ sont connus et $\alpha_i$ a été précédemment évalué par l'algorithme de maximisation du Lagrangien dual $L_D$.
On peut remarquer que, d'après (17) et (18), les seuls points pouvant être à l'extérieur du tube d'insensibilité ($\xi_i,\xi_i^*\neq 0$) sont ceux pour lesquels les multiplicateurs de Lagrange correspondants respectent $\alpha_i=C$ ou $\alpha_i^*=C$. D'autre part, pour les exemples à l'intérieur du tube ($|y_i-f(\boldsymbol{x}_i)|<\varepsilon$ et $\xi_i,\xi_i^*=0$), on a $\alpha_i,\alpha_i^*=0$ pour pouvoir vérifier les conditions (15) et (16). Cela entraîne la propriété de parcimonie (*sparsity*) des SVMs : seuls les exemples avec des $\alpha_i,\alpha_i^*$ correspondants non nuls sont nécessaires pour implémenter le modèle (14). Ces exemples sont appelés Vecteurs de Support ou *Support Vectors* (SVs) et ne constituent dans la pratique qu'une petite fraction des données d'apprentissage [40].

### *B. Problème non-linéaire*

L'extension des SVMs au cas non-linéaire est assez simple et repose sur la projection des données dans un espace de dimension supérieure dans lequel le problème devient linéaire. L'exemple suivant [41] montre l'implémentation d'une fonction quadratique par un SVM linéaire. Soit la projection $\Phi$ définie par

$$\Phi:\mathbb{R}^2\to\mathbb{R}^3,\ \Phi(x_1,x_2)=(x_1^2,\sqrt{2}x_1x_2,x_2^2)\ . \tag{19}$$

L'algorithme SVM précédemment décrit appliqué aux données projetées dans $\mathbb{R}^3$ par $\Phi$ donnerait une fonction linéaire dans $\mathbb{R}^3$ correspondant à une fonction quadratique dans $\mathbb{R}^2$.
Cette approche semble raisonnable pour cet exemple en dimension 2, mais peut vite devenir coûteuse en temps de calculs pour des non-linéarités polynomiales d'ordre supérieur ou des problèmes à grande dimension d'entrée [40]. Pour résoudre ces problèmes, une projection implicite par fonctions noyaux est utilisée.
Le fait que les données $\boldsymbol{x}_i$ n'apparaissent que sous forme de produit scalaire aussi bien dans la formulation du problème d'optimisation (11) que dans la fonction $f$ (14) constitue un des principaux avantages des SVMs. Il n'est pas nécessaire de connaître la projection $\Phi:\mathbb{R}^p\to\mathcal{F}$ explicitement, mais uniquement le résultat $k(\boldsymbol{x}_i,\boldsymbol{x})=\langle \boldsymbol{\Phi}(\boldsymbol{x}_i),\boldsymbol{\Phi}(\boldsymbol{x})\rangle$ du produit scalaire entre les images des points dans l'espace étendu $\mathcal{F}$, aussi appelé *feature space*.
Le problème non-linéaire peut donc être formulé ainsi

$$\max L_D = -\frac{1}{2}\sum_{i,j=1}^{N}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)k(\boldsymbol{x}_i,\boldsymbol{x}_j) - \varepsilon\sum_{i=1}^{N}(\alpha_i+\alpha_i^*)+\sum_{i=1}^{N}y_i(\alpha_i-\alpha_i^*)\ , \tag{20}$$

sous les contraintes

$$\sum_{i=1}^{N}(\alpha_i-\alpha_i^*)=0 \quad \text{et} \quad \alpha_i,\alpha_i^*\in[0,C]\ . \tag{21}$$

De même, les poids sont donnés par

$$\boldsymbol{w}=\sum_{i=1}^{N}(\alpha_i-\alpha_i^*)\boldsymbol{\Phi}(\boldsymbol{x}_i)\ , \tag{22}$$

et le modèle par

$$f(\boldsymbol{x})=\sum_{i=1}^{N}(\alpha_i-\alpha_i^*)k(\boldsymbol{x}_i,\boldsymbol{x})+b\ . \tag{23}$$

La principale différence avec le cas linéaire est que le vecteur des poids $\boldsymbol{w}$ n'est plus exprimé explicitement dans $\mathbb{R}^p$ mais implicitement dans $\mathcal{F}$. De même, l'algorithme cherchera la fonction la plus plate dans l'espace étendu $\mathcal{F}$ et non plus dans l'espace d'entrée $\mathbb{R}^p$. La fonction $k$ est appelée fonction noyau ou noyau (*kernel function* ou *kernel*). Dans la pratique, seule cette fonction est connue, la projection $\Phi$ ne l'est pas.
Les noyaux les plus couramment utilisés pour les SVMs sont les noyaux polynomiaux, sigmoïdaux et à fonction de base radiale (*Radial Basis Function, RBF*) définis ainsi :

- linéaire : $k(\boldsymbol{x},\boldsymbol{x}')=\langle \boldsymbol{x},\boldsymbol{x}'\rangle$ ;
- polynomial : $k(\boldsymbol{x},\boldsymbol{x}')=(\gamma\langle \boldsymbol{x},\boldsymbol{x}'\rangle+c)^d$ ;
- sigmoïdal : $k(\boldsymbol{x},\boldsymbol{x}')=\tanh(\gamma\langle \boldsymbol{x},\boldsymbol{x}'\rangle+c)$ ;

– RBF : $k(\boldsymbol{x},\boldsymbol{x}') = \exp\left(\dfrac{-\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2\sigma^2}\right)$ .

Pour être admissible, une fonction noyau doit vérifier les conditions de Mercer [30] qui se résument à vérifier que la fonction $k : \mathbb{R}^{p\times p} \to \mathbb{R}$ corresponde bien à un produit scalaire définissant une norme dans un espace étendu $\mathcal{F}$.

### C. Programmation linéaire et SVR

Une autre formulation basée sur la minimisation de la norme $\ell_1$ des paramètres conduit à un problème de programmation linéaire et non plus quadratique. L'idée n'est plus de maximiser la platitude de la fonction, mais plutôt de minimiser le nombre de SVs qui influe aussi sur les bornes de l'erreur de généralisation. Une estimée du nombre de SVs et donnée par $\sum(\alpha_i + \alpha_i^*)$. La minimisation de ce terme, aussi appelée "rétrécissement" des paramètres (*shrinkage*), conduit en effet à maximiser le nombre de paramètres nuls dans le modèle (23) et donc à minimiser le nombre de vecteurs de support nécessaires pour le calcul de $f$. Par rapport à l'apprentissage SVM par programmation quadratique, l'apprentissage par programmation linéaire conduit généralement à des modèles utilisant moins de vecteurs de support.

Pour la fonction d'erreur $\varepsilon$-insensible, le problème devient [40]

$$\min \sum_{i=1}^{N}(\alpha_i + \alpha_i^*) + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \tag{24}$$

sous les contraintes

$$\begin{aligned} y_i - \sum_{j=1}^{N}(\alpha_j + \alpha_j^*)k(\boldsymbol{x}_j,\boldsymbol{x}_i) - b &\le \varepsilon + \xi_i \qquad (25)\\ \sum_{j=1}^{N}(\alpha_j + \alpha_j^*)k(\boldsymbol{x}_j,\boldsymbol{x}_i) + b - y_i &\le \varepsilon + \xi_i^* \\ \alpha_i, \alpha_i^*, \xi_i, \xi_i^* &\ge 0 \ , \end{aligned}$$

pour $i = 1,\ldots,N$ . Contrairement au SVR classique, le passage à la formulation duale n'apporte aucune simplification. Cependant ce problème peut se résoudre par des techniques de programmation linéaire habituelles.

Une autre formulation donnée par Mangasarian dans [27] et qui implique moins de variables dans le problème s'écrit

$$\min \sum_{i=1}^{N} a_i + C\sum_{i=1}^{N}\xi_i \ , \tag{26}$$

sous les contraintes

$$\begin{array}{rcl} -\xi_i \le & \sum_{j=1}^{N}\alpha_j k(\boldsymbol{x}_j,\boldsymbol{x}_i) + b - y_i & \le \xi_i \\ 0 \le & \varepsilon & \le \xi_i \\ -a_i \le & \alpha_i & \le a_i,\ i = 1,\ldots,N \ , \end{array} \tag{27}$$

soit, sous forme matricielle :

$$\begin{array}{rcl} \min & \mathbf{1}^T\boldsymbol{a} + C\mathbf{1}^T\boldsymbol{\xi} & \\ \text{s.c.} \quad -\boldsymbol{\xi} \le & \boldsymbol{K}\boldsymbol{\alpha} + b\mathbf{1} - \boldsymbol{y} & \le \boldsymbol{\xi} \\ -\mathbf{0} \le & \varepsilon\mathbf{1} & \le \boldsymbol{\xi} \\ -\boldsymbol{a} \le & \boldsymbol{\alpha} & \le \boldsymbol{a} \ . \end{array} \tag{28}$$

Dans ce cas, le modèle est donné par

$$f(\boldsymbol{x}) = \sum_i \alpha_i k(\boldsymbol{x},\boldsymbol{x}_i) + b = \boldsymbol{K}(\boldsymbol{x},\boldsymbol{X}^T)\boldsymbol{\alpha} + b \ . \tag{29}$$

Sous cette forme, le problème ne requiert pas que le noyau $k$ respecte les conditions de Mercer. La seule contrainte est qu'il soit symétrique [28]. En effet, contrairement au problème quadratique qui minimise une norme des poids induite par le noyau dans le *feature space* $\mathcal{F}$, la minimisation de la norme $\ell_1$ des paramètres est ici réalisée dans $\mathbb{R}^N$. Il n'est donc pas nécessaire de disposer d'un noyau qui défini un produit scalaire et induit une norme dans $\mathcal{F}$.

## III. Questions pratiques

### A. Réglage des hyperparamètres

Dans la méthode SVM, différents hyperparamètres apparaissent : $C$, qui représente le compromis entre la complexité du modèle et l'erreur sur les données d'apprentissage ; $\varepsilon$, qui correspond à la largeur du tube d'insensibilité ; les éventuels paramètres de la fonction noyau $k$ ($\sigma$, $\gamma$,...). Ces hyperparamètres sont en général réglés en fonction d'une estimation de l'erreur de généralisation qui peut être évaluée sur un jeu indépendant de données de validation ou par validation croisée [3]. Cela implique de réaliser l'apprentissage pour différentes valeurs et d'estimer leur performances. Dans le cas d'une estimation de l'erreur de généralisation par validation croisée, cette procédure peut se révéler très coûteuse en temps de calcul.

Une idée permettant d'éviter les temps de calcul trop importants dans la recherche des hyperparamètres optimaux est donnée dans [40]. Si la nouvelle valeur de l'hyperparamètre à tester $C_{nouv}$ est voisine de l'ancienne $C_{anc}$, alors un réajustement des multiplicateurs de Lagrange est utilisé pour l'initialisation du problème d'optimisation :

$$\boldsymbol{\alpha}_{nouv} = \tau\boldsymbol{\alpha}_{anc} \quad \text{et} \quad b_{nouv} = b_{anc} \tag{30}$$

En initialisant ainsi l'algorithme SMO (voir Section III-C), un gain de 95% du temps de calcul peut être obtenu par rapport à un réapprentissage complet [40]. La même méthode peut être appliquée pour les paramètres de la fonction noyau.

D'autres approches permettent d'optimiser le réglage des hyperparamètres. Certaines sont basées sur des méthodes évolutionnistes [17], d'autres sur une recherche par descente de gradient [7][2].

### B. Réglage automatique du tube : $\nu$-SVR

La largeur du tube d'insensibilité $\varepsilon$ peut être réglée en considérant le niveau de bruit contenu dans les données. En effet, celui-ci détermine le niveau d'erreur qui ne sera pas prise en compte par l'algorithme. Si le modèle du bruit était disponible, le problème d'identification se résumerait à appliquer la méthode du maximum de vraisemblance pour obtenir le meilleur estimateur. Mais en pratique, les connaissances sur le bruit sont généralement très limitées.

Il existe néanmoins une technique, appelée $\nu$-SVR, permettant de construire un SVM qui ajuste automatiquement la largeur du tube d'insensibilité. Elle consiste à ajouter un terme dans la formulation du problème d'optimisation pour tenter de minimiser $\varepsilon$. Le problème (3) est donc reformulé ainsi

$$\begin{aligned} \min \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\left(\sum_{i=1}^{N}(\xi_i+\xi_i^*) + N\nu\varepsilon\right) \\ \text{s.c.} \quad & \begin{cases} y_i - \langle \boldsymbol{w}, \boldsymbol{x}_i\rangle - b & \leq \varepsilon + \xi_i \\ \langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b - y_i & \leq \varepsilon + \xi_i^* \ , \ i = 1,\ldots,N \ , \\ \xi_i, \xi_i^* & \geq 0 \end{cases} \end{aligned} \tag{31}$$

avec $\nu > 0$. Le problème dual est donné par

$$\begin{aligned} \max L_D^\nu = & -\frac{1}{2}\sum_{i,j=1}^{N}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)k(\boldsymbol{x}_i,\boldsymbol{x}_j) \\ & + \sum_{i=1}^{N} y_i(\alpha_i-\alpha_i^*) \ , \end{aligned} \tag{32}$$

sous les contraintes

$$\sum_{i=1}^{N}(\alpha_i-\alpha_i^*) = 0, \ \sum_{i=1}^{N}(\alpha_i+\alpha_i^*) \leq C\nu N \text{ et } \alpha_i, \alpha_i^* \in [0, C] \ . \tag{33}$$

Le problème d'optimisation qui règle automatiquement $\varepsilon$ est donc similaire au problème originel avec un terme en moins dans $L_D^\nu$ et une contrainte en plus.
De plus, le paramètre $\nu$ choisi tel que $0 < \nu \leq 1$ peut être interprété à la fois comme le pourcentage d'erreurs à l'extérieur du tube et le pourcentage de SVs à la fin de l'apprentissage comme démontré dans [37]. Dans cet article, le lien avec les estimateurs robustes est aussi réalisé en montrant qu'un changement de la valeur $y_i$ pour un point situé à l'extérieur du tube n'influence pas la solution. Ici, le paramètre $\nu$ peut être aussi relié au point de rupture (*breakdown point* [35]) de l'estimateur robuste correspondant. L'article donne aussi la valeur optimale asymptotique pour $\nu$ dans le cas où le type de distribution (Gaussienne, Laplacienne...) du bruit est connu.
Il est aussi possible d'introduire $\varepsilon$ dans le problème de programmation linéaire (24) pour automatiser le réglage du tube. Cela conduit au problème $\nu$-LPR ($\nu$ Linear Programming Regression) suivant [39]

$$\min \frac{1}{N}\sum_{i=1}^{N}(\alpha_i+\alpha_i^*) + \frac{C}{N}\sum_{i=1}^{N}(\xi_i+\xi_i^*) + C\nu\varepsilon \tag{34}$$

sous les contraintes

$$\begin{aligned} & y_i - \sum_{j=1}^{N}(\alpha_j+\alpha_j^*)k(\boldsymbol{x}_j,\boldsymbol{x}_i) - b \leq \varepsilon + \xi_i \\ & \sum_{j=1}^{N}(\alpha_j+\alpha_j^*)k(\boldsymbol{x}_j,\boldsymbol{x}_i) + b - y_i \leq \varepsilon + \xi_i^* \\ & \alpha_i, \alpha_i^*, \xi_i, \xi_i^*, \varepsilon \geq 0 \ , \end{aligned} \tag{35}$$

pour $i = 1, \ldots, N$ .
Pour la formulation (28) du problème, Mangasarian propose une autre formulation équivalente au problème $\nu$-LPR (où $\nu$ correspond ici à $1-\mu$) [27]

$$\begin{aligned} \min \quad & \tfrac{1}{N}\mathbf{1}^T\boldsymbol{a} + \tfrac{C}{N}\mathbf{1}^T\boldsymbol{\xi} - C\mu\varepsilon \\ \text{s.l.c} \quad & \begin{array}{rcl} -\boldsymbol{\xi} \leq & \boldsymbol{K\alpha} + b\mathbf{1} - \boldsymbol{y} & \leq \boldsymbol{\xi} \\ 0 \leq & \mathbf{1}\varepsilon & \leq \boldsymbol{\xi} \\ -\boldsymbol{a} \leq & \boldsymbol{\alpha} & \leq \boldsymbol{a} \ , \end{array} \end{aligned} \tag{36}$$

où le paramètre $\mu \in [0,1]$ règle le taux d'erreurs supérieures à $\varepsilon$ acceptées par l'algorithme. Le choix $\mu = 0$ correspond à un critère ne faisant intervenir que la valeur absolue des erreurs sans zone d'insensibilité ($\varepsilon = 0$), alors que $\mu = 1$ supprimera l'influence des données ($\varepsilon = \infty$). Ici, l'avantage est de ne faire intervenir que $3N + 2$ variables comparé aux $4N+2$ variables du problème (34)-(35).
Une extension à ces méthodes de réglage automatique du tube peut encore être apportée. Jusque là, la largeur du tube $\varepsilon$ était considérée comme constante. L'étape suivante consiste à considérer que la zone d'insensibilité n'a plus une forme fixe de tube et de la remplacer par un modèle paramètrique [37].

### C. Algorithmes et optimisation

Beaucoup de programmes d'optimisation tels que CPLEX, LOQO, Matlab linprog ou quadprog sont capable de résoudre les problèmes linéaires ou quadratiques issus des SVMs. Néanmoins, la taille des problèmes encouragent les chercheurs de la communauté *machine learning* à développer des algorithmes spécifiques. Des considérations générales sur la formulation quadratique du problème SVM se trouvent dans [21], avec une présentation de certains algorithmes et de certaines méthodes d'optimisation comme le *chunking* qui permet de résoudre le problème en le découpant en plusieurs sous-problèmes. En poussant le *chunking* à sa limite, on obtient l'algorithme SMO (*Sequential Minimal Optimization*) [33] qui considère uniquement deux exemples à chaque itération. Les faiblesses de cet algorithme et des propositions d'améliorations ont été décrites depuis lors dans [22]. Il existe un large éventail de programmes comme SVM$^{light}$ [20], libSVM [6], [16] ou HeroSVM [12], [13] (qui a été testé sur une base de 20 000 000 d'exemples), bien souvent téléchargeables sur Internet, développés dans les dernières années pour augmenter la rapidité de l'algorithme ou pour gérer de grandes bases d'apprentissages. La plupart d'entre eux s'appuient sur la méthode du *chunking* ou implémentent l'algorithme SMO.

### D. Application des SVMs à l'identification

Par souci de simplification, seul est rappelé le prédicteur NARMAX d'un système dynamique à temps discret, de la forme

$$y_k = f(\boldsymbol{\varphi}(k-1)) \ , \tag{37}$$

où $\boldsymbol{\varphi}(k-1)$ représente le vecteur des régresseurs défini par

$$\boldsymbol{\varphi}(k-1) = [u_{k-1}, \ldots, u_{k-n_u},\ y_{k-1}, \ldots, y_{k-n_y},\ e_{k-1}, \ldots, e_{k-n_e}]^T\ , \quad (38)$$

et contenant les observations passées de l'entrée $u$, de la sortie $y$ et de l'erreur de prédiction $e_k = y_k - f(\boldsymbol{\varphi}(k-1))$ . Le problème d'identification consiste ici à déterminer la fonction $f$ de manière à minimiser cette erreur de prédiction.
Pour appliquer la méthode SVM à l'identification, il suffit de définir le vecteur d'entrée $\boldsymbol{x}$ comme le vecteur des régresseurs $\boldsymbol{\varphi}(k-1)$ . La base d'apprentissage est alors simplement construite par

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{\varphi}(0)^T \\ \vdots \\ \boldsymbol{\varphi}(k-1)^T \\ \vdots \\ \boldsymbol{\varphi}(N-1)^T \end{bmatrix}\ , \text{ et } \ \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_k \\ \vdots \\ y_N \end{bmatrix}\ . \quad (39)$$

Pour la forme SVM (23), le prédicteur à un pas à l'instant $k-1$ est ensuite donné par

$$y_{k|k-1} = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) k(\boldsymbol{x}_i, \boldsymbol{\varphi}(k-1)) + b\ , \quad (40)$$

une fois l'apprentissage réalisé. Si la méthode par programmation linéaire (28) est utilisée, le prédicteur est alors donné par (29) et devient

$$y_{k|k-1} = \boldsymbol{K}(\boldsymbol{\varphi}(k-1), \boldsymbol{X}^T)\boldsymbol{\alpha} + b\ . \quad (41)$$

## IV. Discussion et perspectives

Cette Section présente tout d'abord les avantages des SVMs par rapport aux approches plus classiques pour les réseaux de neurones puis une éventuelle extension de la méthode.

### *A. SVMs et réseaux de neurones*

La formulation du modèle (29) est très semblable à la formulation classique de la sortie d'un réseau de neurones [38]

$$y_k = \sum_i \alpha_i g_i(\boldsymbol{\varphi}(k-1))\ , \quad (42)$$

où $\boldsymbol{\varphi}(k-1)$ est le vecteur des régresseurs, $\alpha_i$ les paramètres et $g_i$ les fonctions d'activation du réseau. Par exemple, pour une fonction d'activation RBF (*Radial Basis Function*) de la forme

$$g_i(\boldsymbol{\varphi}(k-1)) = \exp\left(\frac{\|\boldsymbol{\varphi}(k-1) - \boldsymbol{\gamma}_i\|^2}{2\sigma^2}\right)\ , \quad (43)$$

la fonction de sortie du réseau (42) est équivalente, à une constante près, aux fontions de sortie des SVMs (23) ou (29). Bien que les méthodes d'apprentissage restent différentes, il existe une certaine équivalence entre les SVMs et les réseaux régularisés [15].

Cependant, l'apprentissage SVM apporte un certain nombre de réponses aux problèmes rencontrés par l'apprentissage standard des réseaux de neurones :

1. la sélection des centres $\boldsymbol{\gamma}_i$ est automatiquement réalisée par l'algorithme ;
2. l'apprentissage correspond à un problème d'optimisation convexe (unicité de la solution) pour lequel des algorithmes efficaces ont été développés ;
3. la fonction de sortie est douce (*smooth*) (régularisation intrinsèque) ;
4. le problème de la dimension des données (*curse of dimensionality*) est évité grâce aux noyaux ;
5. une bonne généralisation peut être obtenue à partir de peu de données.

### *B. Incorporation de connaissances a priori*

Dans le cadre de l'identification des systèmes, une certaine quantité de connaissances a priori est généralement disponible. Inclure ces connaissances dans l'apprentissage permettrait d'obtenir un meilleur modèle du système. Quelques travaux commencent à s'intéresser au problème d'incorporation de connaissances a priori dans la régression SVM [25], [29]. Cependant, ceux-ci restent rares et appliqués à des types de connaissances bien précis. Cela est certainement dû au fait que la plupart des recherches sur les SVMs sont menées dans le cadre de la reconnaissance des formes. Les connaissances a priori pour ces problèmes sont très souvent du type invariance de la sortie (la classe de l'exemple en l'occurrence) à une transformation de l'entrée. Par exemple, pour la reconnaissance de caractères, l'image d'un '2' reste l'image d'un '2' après rotation ou translation. Or, ce type de connaissances est très rarement disponible dans des problèmes de régression ou d'identification.
Les travaux en cours portent sur l'incorporation de connaissances a priori dans l'apprentissage SVM pour la régression par une méthode générale d'ajout de contraintes dans le programme d'optimisation. Cette méthode permet d'inclure des connaissances exactes ou approximatives, globales ou locales, en particulier dans des régions pauvres en données, comme :

- un modèle a priori du système (issu de lois physiques ou d'un premier apprentissage sur d'autres données) ;
- des connaissances sur les dérivés (modèle de rugosité...) ;
- des points particuliers (ordonnée à l'origine, points d'équilibre, pic d'une fonction...) ;
- des bornes sur la sortie du système ou ses dérivés (maximum local ou global, variation maximale...).

## V. Exemple

Soit le système non-linéaire suivant [8], [4], [5] :

$$\begin{aligned} y_k = &\left(0.8 - 0.5\exp(-y_{k-1}^2)\right) y_{k-1} \\ &- \left(0.3 - 0.9\exp(-y_{k-1}^2)\right) y_{k-2} \\ &+ 0.1\sin(\pi y_{k-1}) + e_k\ , \end{aligned} \quad (44)$$

où $e_k$ représente un bruit gaussien de moyenne nulle. Pour ce système, le vecteur des régresseurs contient deux

TABLE I

NOMBRE DE CENTRES RETENUS PAR LES DIFFÉRENTS ALGORITHMES AVEC L'ERREUR DE TEST CORRESPONDANTE POUR DIFFÉRENTES TAILLES DE LA BASE D'APPRENTISSAGE.

| | | RBF | QP-SVR | | | LP-SVR | | |
|---|---|---|---|---|---|---|---|---|
| | | OLS | $\varepsilon = 0.2$ | $\varepsilon = 0.1$ | $\varepsilon = 0.05$ | $\varepsilon = 0.2$ | $\varepsilon = 0.1$ | $\varepsilon = 0.05$ |
| $N = 100$ | $N_{centres}$ | 11 | 24 | 30 | 31 | 6 | 9 | 9 |
| $\sigma = 1.06$ | $MSE$ ($\times 10^{-3}$) | **0.71** | 0.91 | 1.4 | 0.82 | 3.1 | 0.97 | 0.94 |
| $N = 50$ | $N_{centres}$ | 12 | 12 | 17 | 17 | 7 | 11 | 13 |
| $\sigma = 0.7071$ | $MSE$ ($\times 10^{-3}$) | 2.0 | 2.6 | 2.2 | **1.9** | 8.3 | 2.0 | 2.7 |
| $N = 34$ | $N_{centres}$ | 9 | 9 | 12 | 13 | 6 | 9 | 8 |
| $\sigma = 1.017$ | $MSE$ ($\times 10^{-3}$) | 2.7 | 6.7 | 2.2 | 1.5 | 9.0 | **1.2** | 2.0 |
| $N = 20$ | $N_{centres}$ | 8 | 7 | 9 | 10 | 4 | 6 | 8 |
| $\sigma = 0.9387$ | $MSE$ ($\times 10^{-2}$) | 2.4 | 6.7 | 3.1 | 3.4 | 16 | 2.6 | **1.2** |

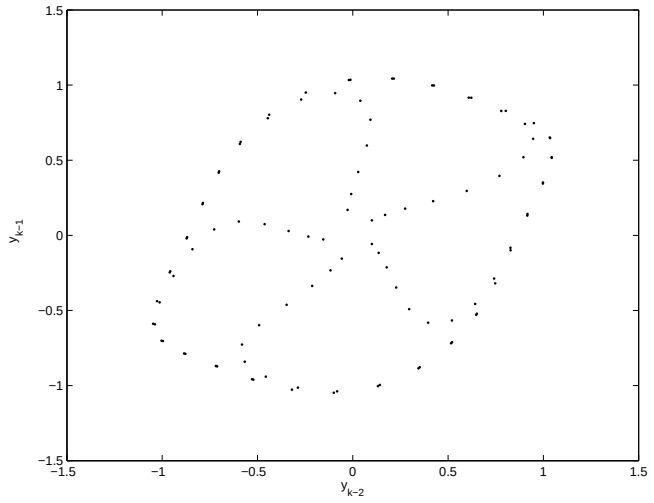


Fig. 3. Trajectoire du système non bruité dans le plan de phases pour les conditions initiales $y_{k-1} = y_{k-2} = 0.1$ .

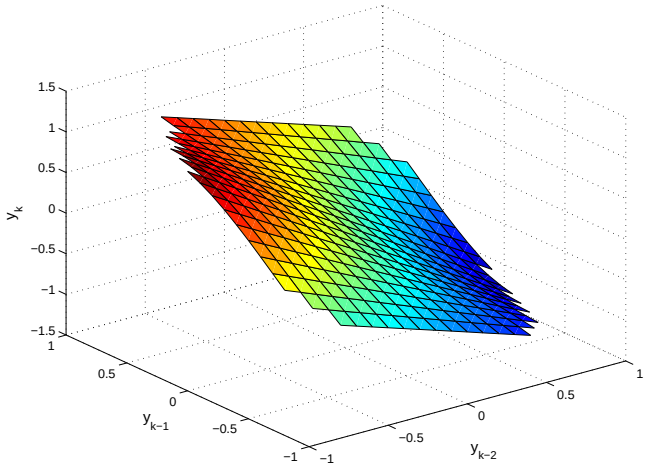


Fig. 4. Surface de régression du système.

sorties retardées : $\varphi(k-1) = [y_{k-1},\ y_{k-2}]$. Ce système possède un point d'équilibre instable à l'origine duquel sa sortie part en spirale vers une courbe invariante fermée comme montré par la trajectoire Figure 3 pour une condition initiale de $y_{k-1} = y_{k-2} = 0.1$ et $e_k \equiv 0$. La surface de régression du système est présentée Figure 4.

Pour montrer la capacité de généralisation à partir de peu d'exemples des SVMs, une trajectoire de 100 points $y_k$ est générée à partir du système (44) pour les conditions initiales $y_{k-1} = y_{k-2} = 0$ et $e_k \sim \mathcal{N}(0, 0.1^2)$. Quatre bases d'apprentissage de tailles $N = 20$, 34, 50 et 100, sont alors construites selon (39) et en prenant un point sur 5, 3, 2 et 1. La comparaison est effectuée entre les SVMs à base de programmation quadratique, QP-SVR (3), les SVMs à base de programmation linéaire, LP-SVR (28), et une méthode de sélection ascendante des centres pour les réseaux RBF connue sous le nom de Orthogonal Least Squares (OLS) [9], avec un critère d'arrêt basé sur une estimation de l'erreur de généralisation [31]. Ces trois algorithmes utilisent des noyaux RBF de largeur $\sigma$ réglée d'après les données selon $\sigma = \sqrt{\max_{i,j} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|/2}$. Le paramètre $C$ pour la méthode QP-SVR est réglé à $C = 6$, 7, 10 et 20 respectivement pour $N = 100$, 50, 34 et 20. La méthode LP-SVR requiert un $C$ plus élevé qui est donc réglé à $C = 20$, 70, 100 et 300 suivant la taille de la base d'apprentissage $N$. Le test est basé sur le nombre de centres (ou vecteurs de support) du modèle et l'erreur de prédiction à un pas évaluée sur $n = 100$ points par $MSE = 1/n \sum_{i=1}^{n} (y_k - y_{k|k-1})^2$, où $y_k$ est la sortie du système (44) non bruité et $y_{k|k-1}$ la sortie du modèle. Le tableau I montre le nombre de centres ainsi que cette erreur de test pour les 3 algorithmes et différents réglages de $\varepsilon$. L'erreur de test la plus faible apparaît en gras, mais le choix du "meilleur" modèle devrait aussi prendre en compte le nombre de centres. La Figure 5 montre les exemples d'apprentissage pour $N = 34$ ainsi que les 9 vecteurs de support retenus par l'apprentissage LP-SVR avec $\varepsilon = 0.1$. On peut voir sur la Figure 6 que le modèle (41) ainsi créé approxime correctement la dynamique du système (44), notamment au niveau du point d'équilibre instable et de la courbe invariante fermée. De même, la surface de régression du système (Fig. 4) est approximée par la surface de réponse du modèle montrée sur la Figure 7. La Figure 8 rend compte de la différence entre ces deux surfaces et donc de l'erreur d'approximation

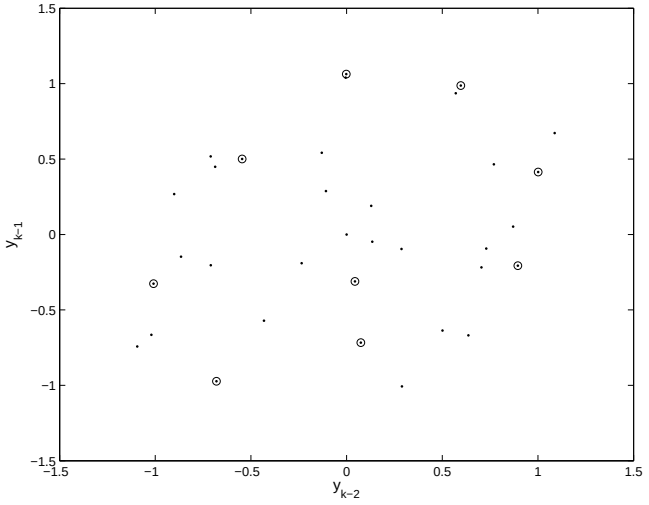


Fig. 5. Trajectoire du système bruité dans le plan de phases constituant la base d'apprentissage ($N = 34$). Les vecteurs de support retenus par l'apprentissage LP-SVR ($\varepsilon = 0.1$) sont entourés.

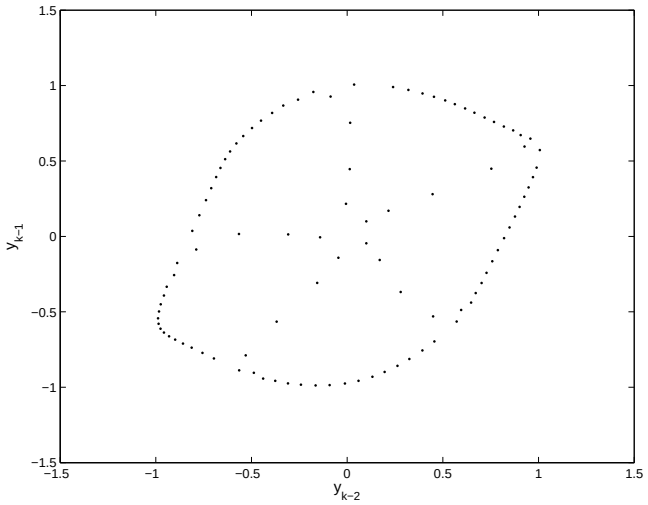


Fig. 6. Trajectoire du modèle LP-SVR ($\varepsilon = 0.1$) dans le plan de phases pour les conditions initiales $y_{k-1} = y_{k-2} = 0.1$ .

qui, sur cet exemple, est toujours inférieure à la taille du tube d'insensibilité $\varepsilon = 0.1$ fixée pour l'apprentissage. Les Figures 9 à 11 montrent les résultats obtenus par l'algorithme OLS pour $N = 34$.

D'après le tableau I, l'algorithme QP-SVR requiert légèrement plus de vecteurs de support (SVs) que le RBF de fonctions de base pour obtenir les mêmes performances. Au contraire, l'algorithme LP-SVR permet de diminuer le nombre de SVs tout en conservant une erreur de test faible. De plus, la trajectoire du modèle RBF en simulation (Fig. 9) montre quelque difficultés à approximer la courbe invariante fermée en comparaison à la trajectoire obtenue par le modèle LP-SVR (Fig. 6). Une simulation sur 1000 points a montré que le modèle RBF ne tend pas vers une courbe invariante fermée mais vers un cycle d'ordre $k = 5$ et donc vers un régime périodique. Pour la même simulation, le modèle LP-SVR parcourt bien la courbe invariante fermée et aucun cycle d'ordre $k$ n'a pu être identifié.

Ces expériences permettent de formuler un certain nombre de remarques :

– la forme LP du problème SVR conduit à une réduction du nombre de vecteurs de support par rapport à la forme QP, comme prévu en Sect. II-C ;
– lorsque le nombre de données est élevé ($N = 100$), la méthode OLS conduit à une erreur plus faible que les méthodes SVM pour un nombre de centres équivalent ;
– en général, la méthode QP-SVR nécessite un nombre de centres légèrement plus élevé que la méthode OLS pour des performances équivalentes ;
– lorsque le nombre de données est faible, la méthode LP-SVR permet d'obtenir une erreur de test plus faible que la méthode OLS pour un nombre de centres identique ou légèrement réduit.

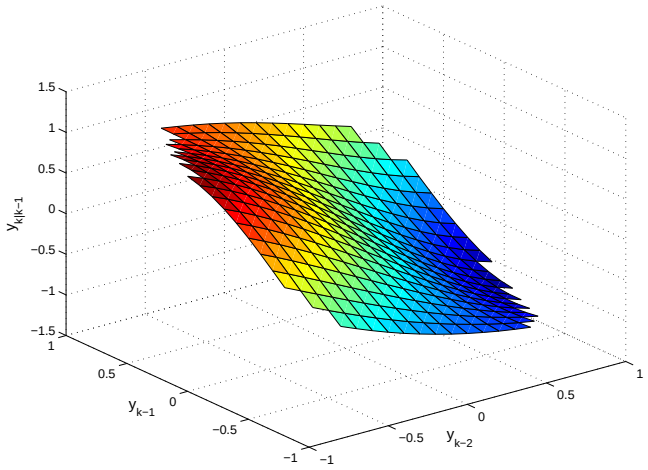


Fig. 7. Surface de réponse du modèle LP-SVR ($\varepsilon = 0.1$).

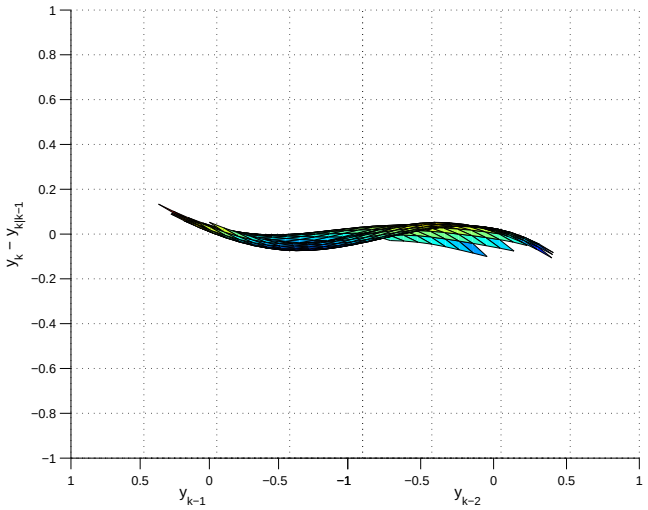


Fig. 8. Erreur entre les surfaces de régression du système (Fig. 4) et de réponse du modèle LP-SVR (Fig. 7).

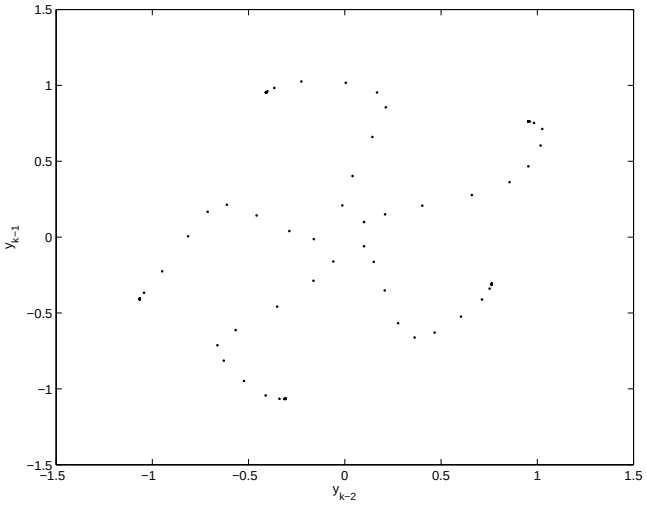


Fig. 9. Trajectoire du modèle RBF dans le plan de phases pour les conditions initiales $y_{k-1} = \ y_{k-2} = 0.1$ .

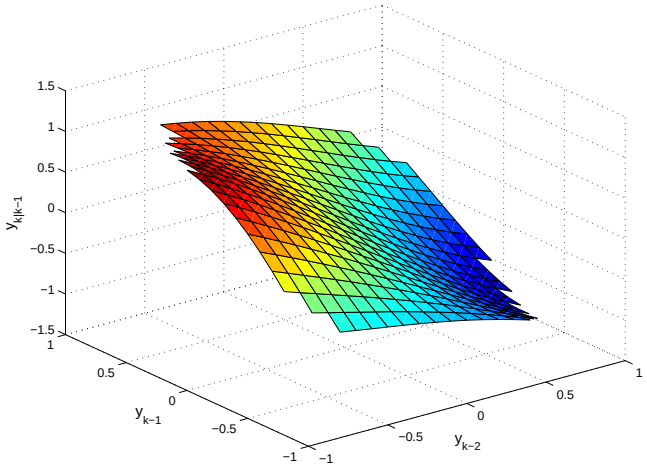


Fig. 10. Surface de réponse du modèle RBF.

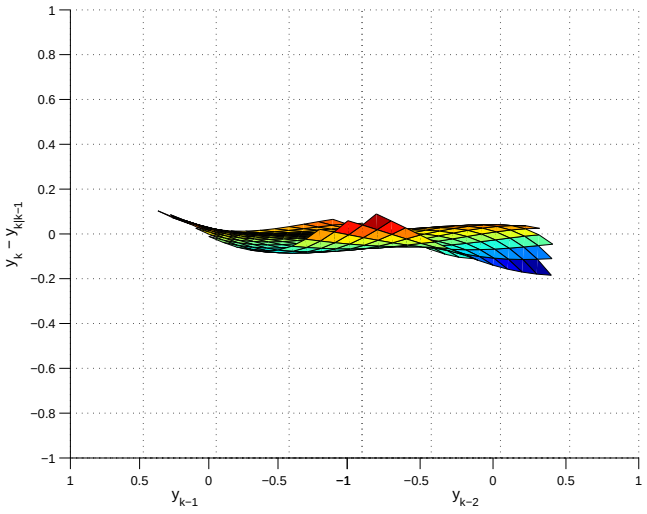


Fig. 11. Erreur entre les surfaces de régression du système (Fig. 4) et de réponse du modèle RBF (Fig. 10).

## VI. Conclusion

Une introduction aux méthodes SVM pour l'identification des systèmes a été présentée. Certains points importants pour l'application pratique des SVMs tels que le réglage des hyperparamètres, l'implémentation algorithmique ou l'application concrète à l'identification ont ensuite été traités. Un exemple numérique a permis de montrer la capacité de généralisation à partir de peu de données des SVMs ainsi que la sélection automatique des centres. Par rapport à la méthode de référence, qui sélectionne les centres de façon ascendante et sur la base d'une estimation de l'erreur de généralisation, la méthode SVM donne des résultats équivalents avec plus de vecteurs de support quand les données sont en nombre suffisant. Quand peu de données sont disponibles, la méthode SVM donne de meilleurs résultats avec moins de centres. Cependant, l'utilisation des SVMs en identification est encore limitée par leur aspect *boîte noire*. Une évolution des SVMs vers un modèle de type *boîte grise* permettant l'incorporation de connaissances a priori dans l'apprentissage augmenterait de manière non négligeable leur intérêt et le panel d'applications éventuelles.

## Références


[1] J.L. An, Z.O. Wang, Q.X. Yang, Z.P. Ma, and C.J. Gao. Study on Method of On-Line Identification for Complex Nonlinear Dynamic System Based on SVM. *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, 3, 2005.

[2] N. E. Ayat, M. Cheriet, and C. Y. Suen. Optimizing the SVM kernels using an empirical error minimization scheme. In S.W. Lee and A. Verri, editors, *Pattern Recognition with Support Vector Machines. Lecture Notes in Computer Science*, volume 2388, pages 354–369, 2002.

[3] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[4] Martin Brown and Chris Harris. *Neurofuzzy Adaptive Modelling and Control*. Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK, 1994.

[5] WC Chan, CW Chan, KC Cheung, and CJ Harris. Modelling of nonlinear dynamical systems using support vector neural networks. *Engineering Applications of Artificial Intelligence*, 14 :105–113, 2001.

[6] C. Chang and C. Lin. LibSVM : a library for support vector machines, 2001. http ://www.csie.ntu.edu.tw/∼cjlin/libsvm/.

[7] O. Chapelle, V. N. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1-3) :131–159, 2002.

[8] S. Chen and S. A. Billings. Neural networks for nonlinear dynamic system modelling and identification. *Int. Journal of Control*, 56 :319–346, 1992.

[9] S. Chen, CFN Cowan, and PM Grant. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Trans. on Neural Networks*, 2(2) :302–309, 1991.

[10] S. Chen, X. Hong, CJ Harris, and X. Wang. Identification of nonlinear systems using generalized kernel models. *Control Systems Technology, IEEE Transactions on*, 13(3) :401–411, 2005.

[11] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[12] J. X. Dong, Adam Krzyzak, and Ching Y. Suen. A fast parallel optimization for training support vector machines. In Petra Perner and Azriel Rosenfeld, editors, *MLDM*, volume 2734 of *Lecture Notes in Computer Science*, pages 96–105. Springer, 2003.

[13] J. X. Dong, Adam Krzyzak, and Ching Y. Suen. Fast SVM training algorithm with decomposition on very large data sets. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(4) :603–618, 2005.

[14] PML Drezet and RF Harrison. Support vector machines for system identification. *Control'98. UKACC International Conference on (Conf. Publ. No. 455)*, 1, 1998.

[15] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13 :1–50, 2000.

[16] R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using the second order information for training SVM. *Journal of Machine Learning Research*, 6 :1889–1918, 2005.

[17] F. Friedrichs and C. Igel. Evolutionary tuning of multiple SVM parameters. In M. Verleysen, editor, *Proc. of the 12th Europ. Symp. on Artificial Neural Networks (ESANN)*, pages 519–524, Bruges, Belgium, 2004.

[18] J. Gao and D. Shi. Sparse kernel regression modelling based on l1 significant vector learning. In *Int. Conf. on Neural Networks and Brain (ICNN&B)*, volume 3, 2005.

[19] Tommi Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In M. J. Kearns, S. A. Solla, and D. A. Cohn, editors, *NIPS*, pages 487–493, Cambridge, MA, USA, 1998. The MIT Press.

[20] T. Joachims. Making large-scale support vector machine learning practical. In Schölkopf et al. [36], pages 169–184.

[21] Linda Kaufman. Solving the quadratic programming problem arising in support vector classification. In Schölkopf et al. [36], pages 147–167.

[22] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation*, 13(3) :637–649, 2001.

[23] F. Lauer, M. Bentoumi, G. Bloch, G. Millerioux, and P. Aknin. Ho-kashyap with early stopping versus soft margin svm for linear classifiers - an application. In Yin et al. [44], pages 524–530.

[24] F. Lauer, C. Y. Suen, and G. Bloch. A trainable feature extractor for handwritten digit recognition. Pattern Recognition, to appear, 2006.

[25] M. Lázaro, I. Santamaria, F. Pérez-Cruz, and A. Artés-Rodriguez. Support vector regression for the simultaneous learning of a multivariate function and its derivatives. *Neurocomputing*, 69 :42–61, 2005.

[26] KL Lee. Time series prediction using support vector machines, the orthogonal and the regularized orthogonal least-squares algorithms. *International Journal of Systems Science*, 33(10) :811–821, 2002.

[27] O. L. Mangasarian and D. R. Musicant. Large scale kernel regression via linear programming. *Machine Learning*, 46(1-3) :255–269, 2002.

[28] O. L. Mangasarian, Jude W. Shavlik, and Edward W. Wild. Knowledge-based kernel approximation. *J. Mach. Learn. Res.*, 5 :1127–1141, 2004.

[29] O. L. Mangasarian and E. W. Wild. Nonlinear knowledge in kernel approximation. Technical Report 05-05, Data Mining Institute, University of Wisconsin, 2005.

[30] J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 209 :415–446, 1909.

[31] Mark Orr. Introduction to radial basis function networks. Technical report, Center for Cognitive Science, University of Edinburgh, Scotland, UK, 1996.

[32] Edgar Osuna, Robert Freund, and Federico Girosi. Training support vector machines : an application to face detection. In *Proc. of the Conf. on Computer Vision and Pattern Recognition, San Juan, Puerto Rico*, pages 130–136, Washington, DC, USA, 1997. IEEE Computer Society.

[33] John C. Platt. Fast training of support vector machines using sequential minimal optimization. In Schölkopf et al. [36], pages 185–208.

[34] M. Pontil and A. Verri. Support vector machines for 3D object recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(6) :637–646, 1998.

[35] P.J. Rousseeuw and A.M. Leroy. *Robust Regression and Outlier Detection*. Wiley-Interscience, 2003.

[36] B. Schölkopf, C. J.C. Burges, and A. J. Smola, editors. *Advances in kernel methods : Support vector learning*, Cambridge, MA, USA, 1999. MIT Press.

[37] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12 :1207–1245, 2000.

[38] J. Sjoberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P.Y. Glorennec, H. Hjalmarsson, and A. Juditsky. Nonlinear black-box modeling in system identification : a unified overview. *Automatica*, 31(12) :1691–1724, 1995.

[39] A. Smola, B. Schölkopf, and G. Rätsch. Linear programs for automatic accuracy control in regression. In *IEEE Conference Publication*, volume 2, pages 575–580, 1999.

[40] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3) :199–222, 2004.

[41] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag, New York, NY, USA, 1995.

[42] Vladimir N. Vapnik. *Statistical learning theory*. John Wiley, New York, NY, USA, 1998.

[43] X.D. Wang and M.Y. Ye. Nonlinear dynamic system identification using least squares support vector machine regression. *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, 2, 2004.

[44] F. Yin, J. Wang, and C. Guo, editors. *Advances in Neural Networks - ISNN 2004, International Symposium on Neural Networks, Dalian, China, August 19-21, 2004, Proceedings, Part I*, volume 3173 of *Lecture Notes in Computer Science*. Springer, 2004.

[45] Li Zhang and Yugeng Xi. Nonlinear system identification based on an improved support vector regression estimator. In Yin et al. [44], pages 586–591.