Utiliser des "support vector machines" pour apprendre un noyau de viabilité

Guillaume Deffuant, Sophie Martin, Laëtitia Chapel

Utiliser des « support vector machines » pour apprendre un noyau de viabilité

Guillaume Deffuant, Sophie Martin, Laetitia Chapel
Cemagref
Laboratoire d'Ingénierie des Systèmes Complexes
24 avenue des Landais
63172 Aubière Cedex
laetitia.chapel@clermont.cemagref.fr

Résumé : Nous proposons d'utiliser un algorithme d'apprentissage particulier, les SVMs, pour résoudre des problèmes de viabilité. Différentes procédures ont déjà été proposées pour approcher un noyau de viabilité mais leur application reste difficile car le résultat est alors un ensemble de points, sans définition explicite. La méthode proposée permet de donner une bonne approximation du noyau de viabilité, alors défini par une expression analytique, ce qui permet d'utiliser une méthode d'optimisation pour trouver le meilleur contrôle à chaque pas. Cette propriété permet également de considérer l'optimisation sur plusieurs pas, ce qui donne une meilleure approximation du noyau. Des premières expériences montrent que l'algorithme approche le noyau de viabilité avec une bonne précision, dans des espaces de dimension 2 à 6.
Mots-clés : Apprentissage, Support Vector Machines, Théorie de la viabilité, Modélisation et commande de systèmes.

1 INTRODUCTION

Les Support Vector Machines (SVMs) sont une méthode de classification qui montre de bonnes performances dans la résolution de problèmes variés tels que la reconnaissance de formes [Burges, 1998] ou la catégorisation de textes [Joachims, 1998]. Le classifieur est alors un hyperplan optimal défini dans un espace « déployé », construit à l'aide d'un nombre réduit de points : les vecteurs de support. Dans cet article, nous montrons comment les SVMs peuvent être utilisées pour résoudre des problèmes de viabilité.

La théorie de la viabilité propose des méthodes et concepts pour contrôler un système dynamique afin de le maintenir dans un ensemble de contraintes de viabilité. L'ensemble des points pour lesquels au moins une solution est viable est appelé noyau de viabilité. Les applications sont nombreuses en économie [Bene, 2001] ou en écologie [Mullon, 2004, Bonneuil, 2003], lorsqu'un système meurt ou se détériore lorsqu'il quitte une certaine zone de l'espace. Cependant, la définition du noyau de viabilité reste difficile : des algorithmes spécifiques ont été développés mais leur application demande un espace mémoire exponentiel avec la dimension de l'espace et le résultat est difficile à manipuler. L'utilisation de SVMs est particulièrement adaptée car le noyau est alors approché par une expression analytique et dépend d'un nombre de points réduit. Cette première propriété permet d'utiliser une méthode d'optimisation afin de trouver un contrôle « optimal » qui permette à un état de rester viable à l'instant suivant. De la même façon, on peut également réaliser une optimisation sur plusieurs pas, ce qui permet d'augmenter la vitesse de convergence de la procédure et d'augmenter la précision de l'approximation du noyau de viabilité.

Nous décrivons une procédure qui permet de donner une approximation du noyau de viabilité en utilisant les SVMs [Deffuant, 2005]. La base d'exemple est alors constituée d'un ensemble de points placés sur une grille, auxquels sont associés un label ($+1$ si le point se situe dans le noyau courant, -1 sinon). Nous décrivons l'algorithme utilisé et des exemples sont donnés pour un système dynamique particulier. Ces résultats montrent que l'algorithme permet de donner une bonne approximation du noyau de viabilité, pour des espaces de 2 à 6 dimensions.

Dans la première partie, nous présenterons les principaux concepts de la théorie de la viabilité. Le principe des SVMs sera rappelé dans la deuxième partie. Nous décrirons ensuite l'algorithme proposé et nous donnerons quelques résultats d'expériences dans différentes dimensions. Pour terminer, nous proposerons quelques pistes de travail.

2 THÉORIE DE LA VIABILITÉ

La théorie de la viabilité [Aubin, 1991] propose des outils et méthodes afin de contrôler un système dynamique dans le but de le garder dans un ensemble d'états admissibles K, appelé ensemble des contraintes.

Considérons un système dynamique défini par son état $x(t) \in \mathbb{R}^n$ et supposons que son évolution $x'(t)$ puisse être modifiée par un contrôle $u(t)$:

$$\begin{cases} x'(t) = \varphi(x(t), u(t)), \text{ pour tout t} \geq 0 \\ u(t) \in U(x(t)) \subset \mathbb{R}^q \end{cases} \quad (1)$$

[Aubin, 1991] définit un état viable dans $K \subset \mathbb{R}^n$ comme un état x_0 pour lequel il existe au moins une fonction de contrôle qui permette à la trajectoire $x(t)$, satisfaisant (1),

de rester indéfiniment dans K (figure 1) :

$$\begin{cases} x(0) = x_0 \\ \forall t \geq 0, x(t) \in K \end{cases} \quad (2)$$

Le problème est de trouver une fonction de contrôle $t \rightarrow u(t)$ qui permette de garder un état viable. Le plus grand sous-ensemble de K, $Viab(K)$, qui contient tous les états $x_0 \in K$ viables est appelé noyau de viabilité :

$$Viab(K) = \{x_0 \in K, \exists u(.), \forall t \geq 0, x(t) \in K\} \quad (3)$$

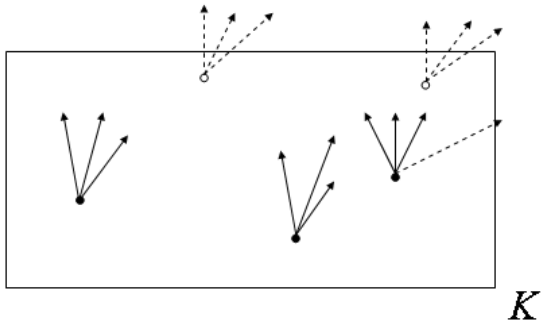

FIG. 1 – États viables à l'instant suivant et états non viables. Les états représentés par un point blanc sont les états non viables (il n'existe aucune trajectoire qui reste dans K à l'instant suivant) et les points noirs sont les états viables (il existe au moins une trajectoire qui reste dans K à l'instant suivant).

Le noyau de viabilité permet de définir directement des politiques de contrôle de viabilité. En effet, il suffit de choisir n'importe quel contrôle qui permette à un état $x(t) \in Viab(K)$ de rester dans le noyau à l'instant suivant (on sait qu'il existe au moins une trajectoire qui permet de garder cet état viable, selon la définition du noyau). Cette procédure permet à n'importe quel état du noyau de rester dans K.
Le noyau de viabilité peut être déterminé de diverses manières grâce à des théorèmes de viabilité [Aubin, 1991], à condition que le système respecte certaines contraintes (systèmes de Marchaud[1] par exemple).
En général, il n'existe pas de définition explicite du noyau de viabilité. Un algorithme spécifique, développé par [Saint-Pierre, 1994] et basé sur les théorèmes de viabilité, permet de calculer une approximation discrète du noyau. Pour cela, il associe au système dynamique (1) un système dynamique discret, défini sur une grille de points, qui approche le problème initial. $Viab(K)$ est alors défini comme le plus grand sous-ensemble E de K tel que :

$$\{\forall x(t) \in E, \exists u(t) \in U(x), x(t+dt) \in E\} \quad (4)$$

On cherche donc le noyau de viabilité comme un ensemble E tel qu'il existe au moins une trajectoire qui permette à un état $x(t)$ de rester dans E à l'instant suivant. En partant de l'ensemble des contraintes K, on enlève progressivement les points pour lesquels aucun contrôle ne permet de rester à l'instant suivant dans l'approximation courante du noyau. [Saint-Pierre, 1994] montre que le noyau de viabilité discret converge vers le vrai noyau lorsque le pas de la grille tend vers 0.
L'algorithme est très rapide mais la taille de la grille croît exponentiellement avec la dimension du problème et la précision nécessaire. De plus, il manipule des ensembles de points, ce qui ne permet pas d'utiliser des techniques d'optimisation classiques. Le but de ce papier est de montrer comment l'utilisation d'un algorithme d'apprentissage, les SVMs en particulier, permet de donner une expression analytique du noyau et de résoudre le problème de viabilité.

[1] La restriction la plus sévère est que, pour chaque état x, l'ensemble des vitesses $\varphi(x,u)$ lorsque $u \in U(x)$ soit convexe

3 SUPPORT VECTOR MACHINES

Le but général de la classification est de construire une fonction qui permet de classer des exemples. Il existe de nombreux algorithmes de classification, les SVMs [Vapnik, 1995] en sont une méthode particulière qui montre de bonnes performances dans de nombreux domaines. Nous rappelons brièvement le principe des SVMs. Pour plus de détail, le lecteur pourra se référer à [Cristianini, 2000] ou [Muller, 2001].

Considérons une base d'exemples $S = \{(x_1,y_1),(x_2,y_2),...,(x_n,y_n)\}$ où x_i est un vecteur dans un espace $\mathscr{X} \in \mathbb{R}^N$ et $y_i \in \{-1,+1\}$. Dans le cas le plus simple, lorsque les données sont linéairement séparables, on construit un hyperplan qui sépare les exemples positifs des exemples négatifs :

$$f(x) = w.x + b = 0 \quad (5)$$

où $w \in \mathbb{R}^N$, $x \in \mathscr{X}$, $b \in \mathbb{R}$ et « . » signifiant produit scalaire.
La propriété remarquable des SVMs est que cet hyperplan est optimal, c'est-à-dire qu'il maximise la distance minimale (marge) entre les exemples et le classifieur. Les points les plus proches, qui seuls sont utilisés pour la détermination de l'hyperplan, sont appelés vecteurs de support (SV) (figure 2).

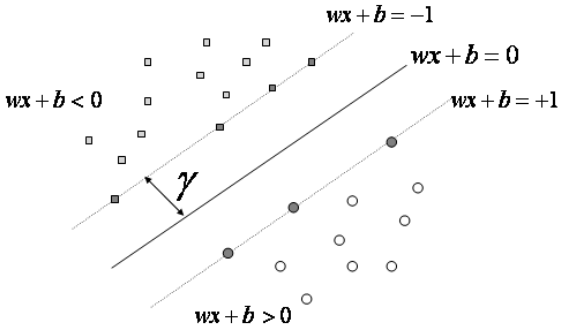

FIG. 2 – Hyperplan séparateur et marge. L'hyperplan séparateur est défini par la fonction $w.x + b = 0$ et la fonction de décision est $signe(w.x + b)$. La marge γ est la distance minimale des points avec l'hyperplan. Les points situés sur les lignes pointillées sont les SV.

Dans le cas non linéairement séparable, on utilise une fonction noyau $K(t,u)$, satisfaisant la condition de Mercer[2], qui permet de projeter les exemples dans un espace de redescription $\mathscr{F}$ de grande dimension et de se ramener ainsi à un cas linéairement séparable.
Il faut souligner que le terme noyau est employé ici avec une signification différente de celle utilisée dans la théorie de la viabilité.
Lorsque les données sont bruitées, on introduit un paramètre de régularisation C qui permet d'autoriser les erreurs de classification.
L'hyperplan séparateur est alors obtenu en résolvant le problème suivant :

$$\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$\text{Sous contraintes :} \quad \begin{cases} 0 \leq \alpha_i \leq C \text{ , } i = 1 \text{ à } n \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases} \tag{6}$$

La résolution de ce problème quadratique permet la définition de la fonction de décision :

$$f(x) = signe(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b) \tag{7}$$

avec $\alpha_i > 0$ SV.
Le calcul de la fonction de décision requiert la résolution d'un gros problème d'optimisation quadratique. Un algorithme spécifique, l'optimisation séquentielle minimale (SMO) [Platt, 1998], a été développé afin de pouvoir manipuler des ensembles de très grande dimension.

4 UTILISATION DES SVMS POUR APPRENDRE UN NOYAU DE VIABILITÉ

4.1 Notations

On utilise une grille finie de pas h, K_h, comme un ensemble d'éléments de K. Les points de cette grille sont notés x_h :

$$\forall x \in K, \exists x_h \in K_h \text{ tel que } \|x - x_h\| < \beta_h \tag{8}$$

avec $\beta_h \rightarrow 0$ quand $h \rightarrow 0$.
On cherche à approcher le noyau de viabilité $Viab_G(K)$ pour le système dynamique suivant :

$$G(x,u) = \{x + \varphi(x,u)dt \text{ pour } u \in U(x)\} \tag{9}$$

On suppose que G est μ-Lipschitz[3].
On considère une fonction $V^n : K_h \rightarrow \{-1, +1\}$ qui associe à chaque étape n le label -1 aux points de la grille x_h qui sortent « franchement » (précisé plus loin) de l'approximation du noyau de viabilité courant et le label $+1$ aux autres points.
La grille K_h est alors divisée en deux sous-ensembles, $V_+^n(K_h)$ et $V_-^n(K_h)$:

$$V_+^n(K_h) = \{x_h \in K_h \text{ tel que } V^n(x_h) = +1\} \tag{10}$$

$$V_-^n(K_h) = \{x_h \in K_h \text{ tel que } V^n(x_h) = -1\} \tag{11}$$

L'utilisation d'un algorithme d'apprentissage comme les SVMs permet d'introduire la fonction $\widehat{V}_n$: $K \rightarrow \{-1, +1\}$ qui, étant donné une grille et les labels associés à chaque point, donne une approximation de V^n. De la même manière, on associe à $\widehat{V}_n$ deux sous-ensembles de K :

$$\widehat{V}_+^n(K) = \left\{x \in K \text{ tel que } \widehat{V}^n(x) = +1\right\} \tag{12}$$

$$\widehat{V}_-^n(K) = \left\{x \in K \text{ tel que } \widehat{V}^n(x) = -1\right\} \tag{13}$$

On peut écrire :

$$\widehat{V}_+^n(K_h) = \widehat{V}_+^n(K) \cap K_h \tag{14}$$

$$\widehat{V}_-^n(K_h) = \widehat{V}_-^n(K) \cap K_h \tag{15}$$

L'utilisation d'un algorithme d'apprentissage permet de considérer l'ensemble des états x de K et non plus uniquement les points x_h de la grille K_h.
Pour tous les états x situés en dehors de l'espace des contraintes K, on pose $\widehat{V}^n(x) = -1$. On note $d(E,F)$ la distance entre les deux ensembles E et F :

$$d(E,F) = \inf\{d(e,f) \ / \ (e,f) \in (E,F)\} \tag{16}$$

4.2 Algorithme utilisé

L'algorithme est basé sur les théorèmes de viabilité. Au départ, on considère que tous les points x_h de K_h sont viables. On supprime ensuite peu à peu les états qui ne sont pas viables à l'instant suivant et on calcule à chaque étape un noyau de viabilité courant. Lorsque les états contenus dans le noyau courant sont tous viables, l'algorithme s'arrête et le noyau courant est alors une approximation du noyau de viabilité.
Les étapes de l'algorithme proposé sont les suivantes :

- Initialisation de la fonction $\widehat{V}^0$:
$$\forall x \in K, \widehat{V}^0(x) = +1 \tag{17}$$
- A l'itération $n+1$, on définit la fonction V^{n+1} à partir de l'approximation courante du noyau $\widehat{V}_+^n(K)$:
$$\begin{array}{l} \forall x_h \in K_h, \\ \quad \text{si } d(G(x_h,u), \widehat{V}_+^n(K)) > \mu\beta(h) \\ \qquad \text{alors } V^{n+1}(x_h) = -1 \\ \qquad \text{sinon } V^{n+1}(x_h) = +1 \end{array} \tag{18}$$
- On obtient la fonction SVM $\widehat{V}^{n+1}$ à partir de la base d'exemples S_h^{n+1} utilisée lors de l'apprentissage :
$$S_h^{n+1} = \left\{(x_h, V^{n+1}(x_h)), x_h \in K_h\right\} \tag{19}$$
- L'algorithme s'arrête lorsque $V_-^{n+1}(K_h) = V_-^n(K_h)$, c'est-à-dire lorsqu'il n'y a plus de modification de labels.

On montre que, sous certaines conditions[4], l'algorithme converge vers le vrai noyau de viabilité en un nombre d'itération n fini.

4.3 Recherche du contrôle optimal grâce à la méthode de descente du gradient

L'utilisation de SVMs permet de définir le noyau courant avec une expression analytique. Supposons que la fonction SVM f est positive lorsque l'on est à l'intérieur du

[2] Une fonction $K(t,u)$ respecte la condition de Mercer si pour toute fonction $g(x)$ telle que $\int g(x)^2 \partial x$ est fini, on a $\int\int K(x,y)g(x)g(y)\partial x \partial y \geq 0$

[3] Une fonction $f : l \rightarrow \mathbb{R}$ est μ-Lipschitz si $\forall (x_1, x_2) \in l^2$, $|f(x_1) - f(x_2)| \leq \mu |x_1 - x_2|$, avec $\mu > 0$.

[4] Par exemple, l'algorithme d'apprentissage ne doit pas faire d'erreur. Pour plus de détails sur ces conditions, le lecteur pourra se référer à [Deffuant, 2005].

noyau de viabilité courant et négative en dehors. On peut alors utiliser une méthode d'optimisation non linéaire afin de trouver le contrôle qui maximise cette fonction. On utilise la méthode de descente de gradient car c'est une méthode courante facile à mettre en oeuvre.

4.3.1 Algorithme de descente de gradient

Le principe de l'algorithme de descente de gradient est de se déplacer dans l'espace dans le sens de la plus grande pente d'une fonction jusqu'à atteindre un minimum. Cette méthode ne garantit que l'obtention d'un minimum local.
Considérons une fonction dérivable $g(x)$. La méthode de descente de gradient recherche le point x^* qui minimise la fonction $g(x)$. Pour cela, on fixe un point initial x_0 et on construit une suite de points $x^{(k)}$ telle que :

$$x^{k+1} = x^k + \eta \nabla g(x^k) \tag{20}$$

avec η pas positif, jusqu'a ce que $\nabla g(x^k)$ soit très petit. On a alors $x^* = x^k$.
Dans le cas où l'on cherche le maximum d'une fonction, il suffit alors de changer le signe de la fonction : maximiser $g(x)$ revient à minimiser $-g(x)$.

4.3.2 Calcul du contrôle optimal dans K

Considérons le système dynamique en temps discret dt suivant :

$$\begin{cases} x(t+dt) = x(t) + \varphi(x(t),u(t))dt \\ u(t) \in U(x(t)) \end{cases} \tag{21}$$

A la première étape de l'algorithme, on cherche un contrôle u^* qui permette de rester dans K à l'instant suivant. Pour cela, on minimise la distance du point $x(t+dt)$ avec le centre C de K. Trouver le meilleur contrôle revient donc à résoudre le problème suivant :

$$\min_{u(t)} \|(x(t) + \varphi(x(t),u(t))dt) - C\|^2 \tag{22}$$

4.3.3 Calcul du contrôle optimal dans la SVM : optimisation à un pas

Considérons la fonction suivante :

$$G(x,u) = x + \varphi(x,u)dt \tag{23}$$

On veut trouver un contrôle qui permet à un état x de rester à l'intérieur du noyau de viabilité courant. Pour cela, on cherche le contrôle u^* qui maximise la fonction $f(G(x,u))$, où f est la fonction SVM courante.

4.3.4 Calcul du contrôle optimal dans la SVM : optimisation à plusieurs pas

Il est possible de réaliser une optimisation sur plusieurs pas à chaque étape de l'algorithme. Pour $n+1$ pas, partant d'un état x_0, le point atteint par la trajectoire est :

$$\begin{aligned} &G_1(x_0,W_1) = G(x_0,u_1) \\ &G_{n+1}(x_0,W_{n+1}) = G(G_n(x_0,W_n),u_{n+1}) \end{aligned} \tag{24}$$

avec $W_1 = (u_1)$, ..., $W_{n+1} = (u_1,u_2,...,u_{n+1})$.
L'ensemble des contrôles optimaux W^*_{n+1} est obtenu en résolvant le problème suivant :

$$\max_{W_{n+1}}(f(G_{n+1}(x_0,W_{n+1}))) \tag{25}$$

5 RÉSULTATS

5.1 Exemple en dimension 2 : modèle simplifié de la croissance d'une population dans un espace limité

5.1.1 Description du modèle

Ce système a été étudié par Maltus puis par Verhulst, et redéveloppé ensuite par [Aubin, 2002] qui a introduit un coefficient d'inertie c dans le système.
Le système représente une population qui a des ressources constantes et pas de prédateurs, mais qui est limitée dans l'espace. L'état $x(t)$ représente alors la population qui grandit ou diminue en fonction d'un taux d'accroissement $y(t)$. La population doit rester dans un certain intervalle $[a,b]$, avec $a > 0$, et le taux d'accroissement $y(t)$ dans $[d,e]$. Le système dynamique en temps discret, avec un intervalle de temps dt, peut être défini de la façon suivante :

$$\begin{cases} x(t+dt) = x(t) + x(t)y(t)dt \\ y(t+dt) = y(t) + u(t)dt \end{cases} \tag{26}$$

avec $u(t) \in [-c,c]$. Le coefficient c permet ainsi de limiter la modification du taux d'évolution à chaque pas de temps dt. On pose $K = [a,b] \times [d,e]$.
Il est possible de dériver analytiquement le noyau de viabilité du problème [Aubin, 2002] :

$$Viab(K) = \{(x,y)\} \text{ tels que}$$

$$\begin{cases} x \in [a,b] \\ y \in [-\sqrt{2c\log(\frac{x}{a})}, \sqrt{2c\log(\frac{b}{x})}] \end{cases} \tag{27}$$

5.1.2 Expériences

Pour qu'un point x_h sorte « franchement » du noyau de viabilité courant, il faut que la distance entre x_h et $\widehat{V}^n_+(K)$ (ou de l'ensemble des contraintes K lors de la première itération) soit supérieure à $\mu\beta(h)$. μ est la constante de Lipschitz du système dynamique (26). Cependant, la convergence de l'algorithme d'approximation est garantie pour des grilles avec un pas h faible. Lorsque l'on travaille avec des grilles qui contiennent moins de points, on implémente un algorithme qui donne une meilleur approximation du noyau de viabilité mais qui offre moins de garantie.
Il semble raisonnable de considérer qu'un point x_h sorte « franchement » du noyau de viabilité courant lorsque $f(x_h) < -1$, même si aucune démonstration ne permet actuellement de le montrer. C'est cette règle qui a été appliquée lors des simulations suivantes. Pour la première itération de l'algorithme, lorsque le noyau de viabilité courant est K, on conserve la distance donnée par le théorème.
On utilise un noyau gaussien pour calculer les différentes fonctions SVMs :

$$K(t,u) = exp\left(-\frac{\|t-u\|^2}{2\sigma^2}\right) \tag{28}$$

En effet, les noyaux gaussiens permettent d'approximer

n'importe quelle fonction de classification et montrent de bonnes performances.
Afin de réaliser les tests, on fixe plusieurs paramètres :

- le pas de la grille h : plus celui-ci est petit, plus l'approximation du noyau de viabilité est fine ;
- le paramètre de régularisation C : il doit être élevé afin de garantir que la SVM ne fasse pas d'erreur ;
- le paramètre σ^2 du noyau gaussien : une forte valeur garantit une SVM plus régulière ;
- le nombre de pas à effectuer à chaque itération : un grand nombre de pas permet à l'algorithme d'effectuer moins d'itérations et de mieux approcher le noyau de viabilité ;
- le paramètre η de l'algorithme de descente de gradient : avec un η faible, la convergence est lente mais on n'est pas sûr de trouver la solution avec un η trop élevé ;
- l'intervalle de temps dt : il doit être fixé en fonction du pas h de la grille et du nombre de pas à réaliser à chaque itération.

Pour les simulations suivantes, on a fixé $a = 0.2$, $b = 3$, $c = 0.5$, $d = -2$, $e = 2$.
Nous avons utilisé la librairie LIBSVM [Chang, 2005], qui implémente un algorithme de type SMO.
La figure 3 présente un exemple de déroulement de l'algorithme, la partie grisée représentant l'approximation du noyau de viabilité à chaque itération et les lignes continues les courbes théoriques. Les paramètres utilisés pour cette simulation sont précisés dans le tableau 1 (simulation 1). On constate qu'avec un pas de la grille élevé (on a seulement 11 points par dimension), l'approximation obtenue est proche du vrai noyau de viabilité : l'écart maximal entre l'approximation et le vrai noyau de viabilité est de l'ordre de $2\beta_h$. 5 apprentissages ont été nécessaires et l'approximation du noyau final nécessite 16 SV.

	Simulation 1	Simulation 2
Dimension	2	2
Pas de la grille h Nombre de points	0.1 121	0.01 10201
dt	0.05	0.005
Nombre de pas	5	5
C	3000	3000
σ^2	0.125	0.125
η	0.9	0.9
Nombre d'itérations	5	42
Nombre de SV	16	105

TAB. 1 – Paramètres et résultats des simulations en dimension 2.

Afin d'obtenir une approximation plus fine du noyau de viabilité, on peut diminuer le pas h de la grille. La figure 4 présente un exemple de résultat. Les paramètres utilisés, le nombre d'itérations nécessaires pour approcher le vrai noyau et le nombre de vecteurs de support sont donnés dans le tableau 1 (simulation 2).

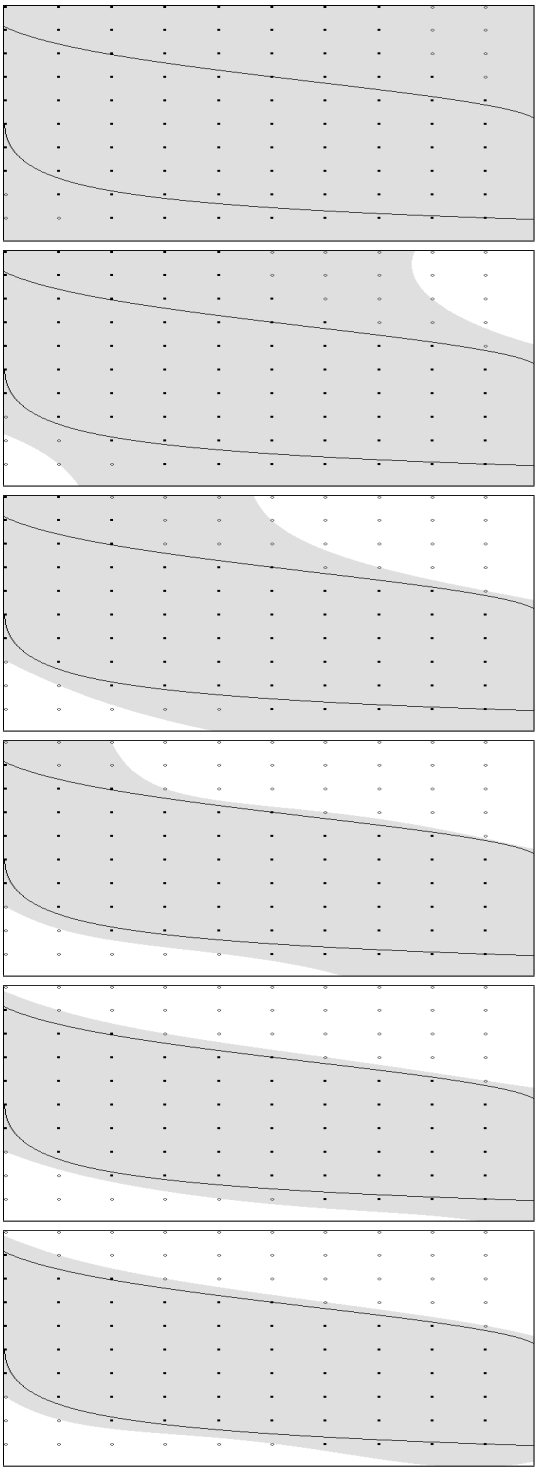

FIG. 3 – Exemple de déroulement de l'algorithme pour le problème population en dimension 2. L'axe des abscisses représente la variable x et celui des ordonnées la variable y. Les points de la grille sont représentés par un rond blanc lorsque le point sort du noyau de viabilité courant et par un carré noir s'il est viable à l'instant suivant. 5 itérations ont été réalisées pour approcher le vrai noyau de viabilité (lignes continues). La partie grisée correspond à l'approximation du noyau courant. La dernière fonction SVM est définie avec 16 vecteurs de support.

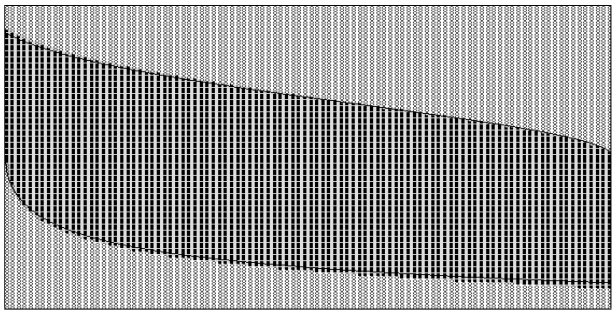

FIG. 4 – Approximation du noyau de viabilité du problème population en dimension 2. La grille contient 10201 points et la fonction SVM est définie avec 105 vecteurs de support.

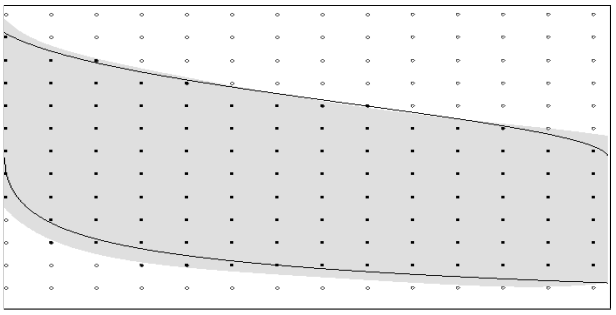

FIG. 5 – Approximation du noyau de viabilité du problème population en dimension 4. L'axe des abscisses représente la variable x_1, les ordonnées la variable y, $x_2 = 1.5$ et $x_3 = 1.5$.

5.2 Exemples en dimensions supérieures

5.2.1 Description du modèle

Le problème de population peut être étendu à un nombre de dimensions supérieur. Dans ce cas, on étudie une population à $n-1$ dimensions et l'on ne tient compte que de la dimension 1, ce qui permet de conserver l'expression analytique du noyau de viabilité.
Le système dynamique peut alors s'écrire de la façon suivante :

$$\begin{cases} x_1(t+dt) = x_1(t) + x_1(t)y(t)dt \\ x_2(t+dt) = x_2(t) \\ \dots \\ x_{n-1}(t+dt) = x_{n-1}(t) \\ y(t+dt) = y(t) + u(t)dt \end{cases} \quad (29)$$

L'intérêt de ce modèle est de pouvoir tester l'algorithme dans des dimensions plus importantes.

5.2.2 Expériences

Des expériences ont été effectuées dans des espaces de dimension 3 à 6. Le tableau 2 détaille les paramètres et les résultats obtenus pour les différentes simulations. La figure 5 donne un exemple de noyau de viabilité obtenu pour un espace de dimension 4.

Dimension	3	4	5	6
h	0.05	0.075	0.1	0.14
Nb de points	9261	38416	161051	262144
dt	0.025	0.0375	0.05	0.075
Nb de pas	4	4	4	3
C	3000	3000	3000	1000
σ^2	0.25	0.25	0.25	0.5
η	0.9	0.9	0.9	0.9
Nb d'itérations	18	31	14	17
Nb de SV	72	313	972	2124

TAB. 2 – Paramètres et résultats de simulations dans les espaces de 3 à 6 dimensions.

La taille de la grille augmente exponentiellement avec le nombre de dimensions. Ainsi, il est difficile de réaliser des expériences dans des espaces de dimensions supérieures à 6 avec la méthode utilisée. Des astuces de calcul doivent être mises en oeuvre afin de pallier ces limitations.

6 DISCUSSION ET PERSPECTIVES

Un nouvel algorithme d'approximation d'un noyau de viabilité a été proposé. Il met en oeuvre une méthode d'apprentissage particulière : les SVMs. La base d'exemples est alors l'ensemble des points d'une grille, auxquels sont associés un label : $+1$ si le point est à l'intérieur du noyau courant, -1 sinon. Des tests dans des espaces de 2 à 6 dimensions montrent que l'approximation donnée par la méthode proposée est proche du vrai noyau de viabilité, même avec un nombre de points par dimension peu important. Des comparaisons effectuées avec des résultats obtenus grâce à l'algorithme de [Saint-Pierre, 1994] montrent que l'approximation donnée par l'algorithme utilisant les SVMs est plus précise, surtout lorsque le nombre de points de la grille est faible. Cependant, il reste difficile de résoudre des problèmes dans des espaces supérieurs à 6 dimensions avec une bonne précision car la taille de la grille explose alors : le temps de résolution de l'algorithme SMO devient très important et le nombre d'optimisations pour trouver un contrôle viable à l'instant suivant explose également.
Ces différentes remarques ouvrent de nouvelles perspectives de travail. Il serait intéressant de ne sélectionner que les points pertinents pour le calcul des différentes SVMs, c'est-à-dire les points qui sont susceptibles d'être des vecteurs de supports, et quelques points plus éloignés, afin de contraindre la fonction SVM dans K. De plus, une meilleure gestion de la grille de points peut être envisagée : utiliser une grille à résolution variable pourrait permettre de travailler dans des espaces de plus grande dimension.

BIBLIOGRAPHIE

[Aubin, 1991] Aubin J.-P., *Viability theory*. Birkhäuser.

[Aubin, 2002] Aubin J.-P., Elements of viability theory for the analysis of dynamic economics. *Ecole thématique du CNRS « Economie Cognitive »*.

[Bene, 2001] Bene C., Doyen L. et Gabay D., A viability analysis for a bio-economic model. *Ecological Economics*, 36(3) :385–396.

[Bonneuil, 2003] Bonneuil N., Making ecosystem models viable. *Bulletin of Mathematical Biology*, 65(6) :1081–1094.

[Burges, 1998] Burges C. J. C., A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2) :121–167.

[Chang, 2005] Chang C.-C. et Lin C.-J., LIBSVM : a library for support vector machines (version 2.8). Disponible sur « http ://www.csie.ntu.edu.tw/ cjlin/libsvm/ ».

[Cristianini, 2000] Cristianini N. et Shawe-Taylor J., *Support Vector Machines and other kernel-based learning methods*. Cambridge University Press.

[Deffuant, 2005] Deffuant G., Martin S. et Chapel L., Approximation de noyau de viabilité à l'aide de svms. Rapport technique, Cemagref.

[Joachims, 1998] Joachims T., Text categorization with support vector machines : learning with many relevant features. *Proceedings of ECML-98, 10th European Conference on Machine Learning*.

[Muller, 2001] Muller K.-R., Mika S., Ratsch G., Tsuda K. et Schlkopf B., An introduction to kernel-based learning algorithms. *IEEE Neural Networks*, 12(2) :181–201.

[Mullon, 2004] Mullon C., Curry P. et Shannon L., Viability model of trophic interactions in marine ecosystems. *Nat. Resource Modeling*, 17(1) :27–58.

[Platt, 1998] Platt J.-C., Fast training of support vector machines using sequential minimal optimization. Rapport technique, 98-14, Microsoft Research, Redmond, Washington.

[Saint-Pierre, 1994] Saint-Pierre P., Approximation of viability kernel. *Applied Mathematics & Optimisation*, 29 :187–209.

[Vapnik, 1995] Vapnik V., *The nature of statistical learning theory*. Springer Verlag.