Comparaison experimentale d'algorithmes de regression pour l'apprentissage de modèles cinematiques du robot humanoïde iCub

Alain Droniou, Serena Ivaldi, Olivier Sigaud

Comparaison expérimentale d'algorithmes de régression pour l'apprentissage de modèles cinématiques du robot humanoïde iCub

Alain Droniou, Serena Ivaldi et Olivier Sigaud

Université Pierre et Marie Curie, Institut des Systèmes Intelligents et de Robotique - CNRS UMR 7222, Pyramide Tour 55 - Boîte Courrier 173, 4 Place Jussieu, 75252 Paris CEDEX 5, France,
Contact: prenom.nom@isir.upmc.fr

Résumé :
La complexité croissante des tâches abordées par la robotique humanoïde nécessite des modèles mécaniques précis difficiles à obtenir en pratique. Une approche possible est de laisser le robot acquérir ses propres modèles par apprentissage. Dans Sicard *et al.* (2011), deux algorithmes ont été comparés pour l'apprentissage de modèles cinématiques sur le robot iCub : XCSF et LWPR. Cette comparaison a été menée sur des données issues de simulations. Dans ce travail, nous nous proposons d'étendre cette étude au cas de données provenant du robot iCub afin d'analyser le comportement de ces algorithmes face à des données fortement bruitées dans des conditions réelles d'utilisation. Nous ajoutons également l'étude d'un troisième algorithme, iRFRLS. Après une étude détaillée du paramétrage des trois algorithmes, nous montrons que les résultats obtenus dans Sicard *et al.* (2011) sont toujours valables sur des données réelles : XCSF converge plus lentement, mais vers une erreur plus faible que LWPR. Toutefois, nous montrons que ces deux algorithmes sont surpassés par iRFRLS.
Mots-clés : Apprentissage robotique, Régression incrémentale, iCub

1 Introduction

Au cours des dernières décennies, le contrôle des robots reposait sur des modèles mécaniques analytiques fournis a priori. Cette approche peut être utilisée pour des robots complexes possédant de nombreux degrés de liberté (Pattacini *et al.* (2010)), mais elle ne prend pas en compte les perturbations, comme les frottements solides (qui varient avec l'usure du robot) ou les contacts avec des objets inconnus. Par exemple, lors de l'interaction avec de nouveaux objets, les robots autonomes requièrent de nouveaux modèles

mécaniques pour adapter leur comportement : manipuler une bouteille est différent selon sa taille, son matériau, si elle est vide ou remplie... Les techniques d'apprentissage doivent permettre une généralisation rapide et efficace pour transférer les connaissances entre des situations proches.

Dans Sicard *et al.* (2011), deux algorithmes, LWPR et XCSF, étaient comparés sur l'apprentissage du modèle cinématique du robot humanoïde iCub, en simulation. Afin d'introduire des incertitudes et se placer dans une situation pour laquelle le modèle est inconnu, l'organe terminal était modélisé par une balle verte fixée au bout d'un bâton tenu par le robot. Dans cet article, nous approfondissons l'étude avec des données enregistrées sur le vrai robot. En effet, dès que l'on se place en conditions réelles, il faut gérer de multiples sources de bruit. Dans ces conditions, un bon algorithme doit être robuste et ses capacités de généralisation sont cruciales.

Nous décrivons tout d'abord le problème d'apprentissage ainsi que les algorithmes considérés dans la section 2. En plus d'XCSF et de LWPR, nous étudions un troisième algorithme, iRFRLS, et nous soulignons ses différences par rapport aux deux premiers. Nous présentons ensuite le cadre expérimental dans la partie 3. Dans la partie 4, nous comparons les performances des trois algorithmes et montrons qu'iRFRLS converge plus rapidement et vers une erreur plus faible que les deux autres algorithmes. Nous analysons et discutons les résultats ainsi que l'influence des paramètres dans la partie 5.

2 Cadre théorique

Nous débutons cette étude par quelques rappels théoriques de mécanique, ainsi que par la description des algorithmes étudiés. Notre approche consiste à apprendre le modèle cinématique direct du robot avant de l'inverser pour le contrôle, ce qui permet une plus grande flexibilité que l'apprentissage du modèle inverse (Salaun *et al.* (2009)).

2.1 Modèles cinématiques

Le modèle le plus simple d'un robot est son modèle géométrique, i.e. la relation entre les positions articulaires $\mathbf{q}$ et les coordonnées ξ d'un organe terminal dans un espace des tâches (souvent un espace cartésien) : $\xi = f(\mathbf{q})$.

Lorsque la dimension de l'espace articulaire est supérieure à celle de l'espace des tâches, le système est redondant et il n'y a pas de méthode simple pour calculer l'ensemble des solutions au niveau géométrique. C'est pourquoi

le modèle est généralement dérivé pour obtenir le modèle cinématique, à travers la matrice jacobienne cinématique $J(\mathbf{q}) = \frac{\partial f}{\partial \mathbf{q}}$, qui relie les vitesses opérationnelles aux vitesses articulaires : $\dot{\xi} = J(\mathbf{q})\dot{\mathbf{q}}$.

Afin de contrôler le robot, il est alors nécessaire d'inverser l'équation précédente. On obtient $\dot{\mathbf{q}} = J^{\sharp}(\mathbf{q})\dot{\xi}$ où $J^{\sharp}$ est une pseudo-inverse de J. Il est alors possible de calculer les vitesses articulaires correspondant à la vitesse opérationnelle désirée. Cependant, lorsque le système est redondant et en dehors des singularités, il y a une infinité de solutions $\dot{\mathbf{q}}$ permettant d'obtenir la vitesse désirée $\dot{\xi}^{\star}$. Parmi toutes ces possibilités, nous utilisons la *Damped Least Square Pseudo-inverse* (DLS - PINV) (Deo & Walker (1995)). De plus amples informations sur les méthodes de contrôle utilisées lors de nos expériences sont disponibles dans Sicard *et al.* (2011).

2.2 Algorithmes de régression

Apprendre le modèle mécanique d'un robot peut-être vu comme un problème d'approximation de fonction continue, ce qui constitue en apprentissage supervisé un problème de régression. Plus précisément, les contraintes de la robotique nous conduisent à nous tourner vers des méthodes de régression incrémentale capables de fournir rapidement une solution dans des espaces possédant entre 10 et 100 dimensions. Nous concentrons notre étude sur trois algorithmes de régression incrémentale : LWPR (Vijayakumar & Schaal (2000)), XCSF (Wilson (2001)) et iRFRLS (Gijsberts & Metta (2011)). Ce choix s'appuie sur la revue des différents algorithmes existants menée dans Sigaud *et al.* (2011).

2.2.1 LWPR

LWPR [1] (*Locally Weighted Projection Regression*) (Vijayakumar & Schaal (2000)) est un algorithme d'approximation de fonction incrémental performant dans des espaces de très grande dimension pour un coût computationnel réduit. Il consiste en une combinaison de modèles linéaires pondérés par des gaussiennes, aussi appelées *Receptive Fields* (RF), qui définissent la localité de chaque modèle. Les RF ainsi que les modèles linéaires sont calculés de manière incrémentale à partir des données d'apprentissage.

1. Nous utilisons la bibliothèque C/C++ *liblwpr* disponible sur `http://www.ipab.inf.ed.ac.uk/slmc/software/lwpr/index.html`.

Afin de pouvoir travailler efficacement dans des espaces de grande dimension, les entrées sont projetées sur des espaces de plus faible dimension propres à chaque RF à l'aide de la méthode des moindres carrés partiels (Wold (1975)). Cette technique permet d'extraire les dimensions pour lesquelles la corrélation entre entrées et sorties est maximale.

La prédiction finale est alors donnée par la combinaison de tous les modèles locaux $\hat{y}_i$: $\hat{y}(x) = \frac{\sum_{k=1}^{K} w_k \hat{y}_k(x)}{\sum_{k=1}^{K} w_k}$ où K est le nombre de RF.

Une présentation plus détaillée de la version incrémentale de l'algorithme peut être trouvée dans Schaal *et al.* (2002) ou Vijayakumar & Schaal (2000).

2.2.2 XCSF

XCSF [2] (Wilson (2001)) est un autre algorithme d'approximation de fonctions qui partage quelques points communs avec LWPR, mais qui fait partie de la famille des *Learning Classifier Systems* (LCS) (Holland (1992)). En tant que LCS, XCSF s'appuie sur une population de règles, appelées *classeurs*. Ces classeurs contiennent une partie condition et une partie prédiction. Pour XCSF, la partie condition définit la localité $\phi_i(z)$ du modèle local tandis que la partie prédiction contient le modèle local β_i lui-même. Chaque classeur prédit un vecteur y_i pour chaque entrée x_i. Les modèles β_i sont calculés par une méthode de moindres carrés récursifs (Kailath *et al.* (2000)). Les classeurs forment une population P qui pave la partie condition de modèles superposés.

XCSF n'utilise qu'un sous-ensemble des classeurs pour chaque estimation. En effet, à chaque itération, XCSF calcule l'ensemble de couverture M qui ne contient que les classeurs dont la partie condition $\mathcal{Z}$ correspond aux données en entrée z, i.e. pour lesquels $\phi_i(z)$ est supérieur à un seuil ϕ_0.

L'estimation finale $\hat{y}$ est calculée pour chaque paire (x, z) comme la somme des modèles linéaires de l'ensemble de couverture, pondérés par leurs poids relatifs F définis selon la performance en prédiction de chaque classifieur (*fitness*) : $\hat{y}(x, z) = \frac{\sum_{k=1}^{n_M} F_k(z)\, \hat{y}_k(x)}{\sum_{k=1}^{n_M} F_k(z)}$ où n_M est le nombre de classeurs dans l'ensemble de couverture. L'évolution de la population de classeurs est dirigée par un algorithme génétique hérité de XCS (Wilson (1995)).

Un processus important dans le cadre de cette étude est la condensation. XCSF génère en effet un grand nombre de régions qui se superposent. Pour

2. Nous utilisons le serveur Java *XCSFServer* développé par M. Butz et P. Stalph (Stalph & Butz (2010)).

réduire la taille de la population, la condensation repose sur une méthode de *Closest Classifier Matching* (Butz & Herbort (2008)) dont le but est d'obtenir des ensembles de couverture de taille fixe pour chaque condition z.

2.2.3 iRFRLS

iRFRLS [3] repose sur une méthode de moindres carrés régularisés couplée à des estimateurs aléatoires (Gijsberts & Metta (2011)). Dans cet article, nous présentons l'algorithme d'un point de vue reposant sur la transformée de Fourier.

Quasiment n'importe quelle fonction usuelle peut être calculée par inversion de sa transformée de Fourier. Estimer cette dernière plutôt que la fonction elle-même permet d'exploiter certaines propriétés de régularité pour rendre l'apprentissage plus robuste et plus simple. Puisque nous travaillons sur des fonctions réelles continues définies sur des intervalles finis (et donc virtuellement périodiques), les sommes partielles de la série de Fourier de f,

$$S_n(f(x)) = \frac{a_0(f)}{2} + \sum_{k=1}^{n} a_k(f)\cos\left(kx\frac{2\pi}{T}\right) + \sum_{k=1}^{n} b_k(f)\sin\left(kx\frac{2\pi}{T}\right) \quad (1)$$

convergent vers f quand $n \to \infty$, avec $a_n(f) = \frac{2}{T}\int_{-T/2}^{T/2} f(t)\cos\left(nt\frac{2\pi}{T}\right)\,\mathrm{d}t$ et $b_n(f) = \frac{2}{T}\int_{-T/2}^{T/2} f(t)\sin\left(nt\frac{2\pi}{T}\right)\,\mathrm{d}t$

Pour estimer la fonction, il est possible de considérer la plupart des coefficients comme nuls et d'estimer les autres. La qualité des prédictions dépend alors dans ce cas du nombre D de coefficients (estimateurs) appris. En notant $x_1, ..., x_k$ les points de l'ensemble d'apprentissage et

$$\begin{aligned} f &= [f(x_1), ..., f(x_k)]\,,\ w = [a_{i_0}, b_{i_0}, ..., a_{i_D}, b_{i_D}]^T \\ z(x) &= [cos(\omega_{i_0}x), sin(\omega_{i_0}x), ..., cos(\omega_{i_D}x), sin(\omega_{i_D}x)]^T \\ Z &= [z(x_1), ..., z(x_k)]^T \end{aligned}$$

la prédiction s'écrit alors $\hat{f}(x) = w^T z(x)$ et une méthode usuelle de *support vector regression* (Drucker *et al.* (1997)) peut être utilisée pour estimer les coefficients. La solution est donnée par $w = (\lambda I + Z^T Z)^{-1} Z^T f$ où λ est un terme de régularisation. Une décomposition de Cholesky de $(\lambda I + Z^T Z)$

3. Nous utilisons la bibliothèque C++ développée par A. Gijsberts, disponible dans le dépôt svn officiel iCub `https://robotcub.svn.sourceforge.net/svnroot/robotcub/trunk/iCub`.

permet une inversion facile et un calcul incrémental de w (Björck (1996); Sayed (2008); Gijsberts & Metta (2011)).

iRFRLS ne requiert donc le paramétrage que de trois paramètres : le terme de régularisation λ, le nombre d'estimateurs D et la distribution selon laquelle sont tirées les pulsations ω. À cause d'un fort lien avec les méthodes à noyaux gaussiens, on choisit souvent des distributions normales pour lesquelles le seul paramètre est la variance γ. Dans ces conditions, iRFRLS se comporte comme un algorithme de régression à noyaux gaussiens, et $z(x)$ fournit une approximation des valeurs du noyau (Rahimi & Recht (2008)).

3 Cadre expérimental

Nous utilisons le robot humanoïde iCub, qui mesure 104cm et possède 53 degrés de liberté (Metta *et al.* (2008)). Dans Sicard *et al.* (2011), les algorithmes étaient testés sur un simulateur. Pour cette étude, nous utilisons le vrai robot, nous plaçant ainsi dans des conditions réelles d'utilisation. En conséquence, nous sommes confrontés à de nombreuses sources de bruit : imprécision des capteurs, bruit d'alimentation, vibrations mécaniques et jeux au niveau des articulations, algorithmes de vision peu précis, calibration imparfaite et non simultanéité des mesures.

3.1 Paramétrage des algorithmes

Un des éléments cruciaux pour les méthodes d'apprentissage est le paramétrage des algorithmes. Afin de mener une comparaison objective, nous utilisons une méthode de recherche exhaustive : pour chaque paramètre, nous choisissons un ensemble de valeurs (tables 1(a), 1(b), 1(c)) et nous testons toutes les combinaisons possibles.

Afin d'obtenir des résultats statistiquement significatifs, en utilisant un test de Student à variables appariées (Ruxton (2006)), nous répétons chaque expérience 30 fois sur des ensembles différents. Les ensembles utilisés sont identiques pour les trois algorithmes. La performance est mesurée à partir de l'erreur quadratique moyenne (MSE) sur la prédiction de $\dot{\xi}$. Les ensembles de test sont constitués de 1000 points, distincts des ensembles d'apprentissage.

Grâce à la distinction entre espaces de condition et de prédiction, XCSF permet d'apprendre directement la matrice jacobienne cinématique. En revanche, LWPR et iRFRLS ne permettent pas d'apprendre directement cette ma-

trice. Nous les utilisons pour apprendre le modèle géométrique, que nous dérivons pour obtenir la matrice jacobienne cinématique.

3.2 Données

Nous paramétrons les algorithmes à partir de données enregistrées sur le robot selon le protocole décrit dans Sicard *et al.* (2011). Pour ces expériences, iCub doit atteindre une cible qui se déplace selon une astérisque. Pour ne pas introduire de problématiques complexes de vision, l'organe terminal est représenté par une balle verte fixée au bout d'un bâton tenu en main par iCub. La cible est quant à elle modélisée par une balle rouge (cf. Sicard *et al.* (2011) pour plus de détails). En conséquence, le modèle géométrique est inconnu. Au cours de ces expériences, 6 degrés de liberté sont utilisés : 2 pour la tête et 4 pour le bras. La région explorée est assez réduite : il s'agit d'un cube d'une dizaine de centimètres de côté. Comparé au bras d'iCub, qui mesure une quarantaine de centimètres, cela représente en première approximation 20% de l'espace « utile » atteignable par le robot. Ceci peut biaiser les résultats en renforçant certaines linéarités. À l'issue de ces expériences, nous disposons d'une base de 4 millions de points $(\mathbf{q}, \dot{\mathbf{q}}, \xi, \dot{\xi})$ échantillonnés à environ 20Hz.

4 Résultats

Dans cette partie, nous menons des expériences pour étudier l'influence de chaque paramètre ainsi que l'influence de l'ordre de présentation des données. Nous nous concentrons sur la convergence et le temps de calcul.

4.1 Influence des paramètres

Nous étudions dans cette partie l'influence des paramètres de chaque algorithme. Pour cela, nous calculons l'erreur moyenne obtenue sur toutes les simulations, en fixant la valeur d'un seul paramètre à la fois.

Les 28 paramètres d'XCSF sont trop nombreux pour être testés de manière exhaustive. Nous en sélectionnons 6 identiques à ceux retenus dans Sicard *et al.* (2011). Les résultats sont présentés table 1(a). De même pour LWPR, table 1(b). Puisqu'iRFRLS n'a que trois paramètres, il est possible de mener une recherche exhaustive (table 1(c)). Les courbes figure 1(a) représentent les meilleures et les pires performances obtenues pour les trois algorithmes.

TABLE 1: Influence des paramètres pour (a) XCSF, (b) LWPR et (c) iRFRLS. La notation $v_1 \overset{c}{>} v_2$ signifie que la valeur v_1 du paramètre conduit à de meilleures performances que la valeur v_2, avec une confiance de $c\%$. Une valeur en gras (italique) correspond au meilleur (pire) jeu de paramètres. Le temps de paramétrage correspond au temps nécessaire pour tester tous les jeux de paramètres sur les 30 ensembles de données, sans parallélisation.

(a) XCSF

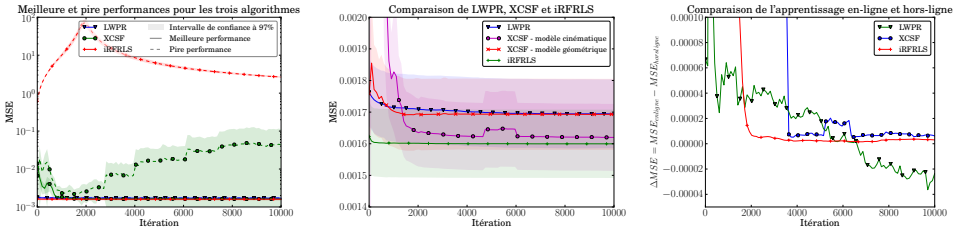
XCSF		
Paramètre	Classement	Temps de calcul
Δ	$0.01 \overset{76}{>} 0.05 \overset{99.2}{>} \boldsymbol{0.1} \overset{84}{>} 0.5$	constant
converConditionRange	$\boldsymbol{0.995} \overset{99.9}{>} 0.7$	constant
minConditionStretch	$0.001 \overset{98}{\approx} \boldsymbol{0.005} \overset{81}{\approx} 0.1$	constant
ϵ_0	$0.01 \overset{98}{>} \mathbf{0.005} \overset{99.4}{>} \textit{0.001}$	constant
maxPopulationSize	$\boldsymbol{1500} \overset{90}{>} 500 \overset{100}{>} 100$	croissant
startCondensation	$\boldsymbol{500} \overset{67}{\approx} 1000 \overset{99.9}{>} 2000 \overset{99.9}{>} 5000$	croissant
Temps de paramétrage		9 jours

(b) LWPR

LWPR		
Paramètre	Classement	Temps de calcul
useMeta	***true*** $\overset{89}{\approx}$ false	constant
updateD	false $\overset{99.7}{>}$ ***true***	*true* plus long
penalty	$\mathbf{0.001} \overset{67}{>} \textit{0.01} \overset{93}{\approx} 0.1$	constant
wGen	$0.9 \overset{92}{\approx} \mathbf{0.2} \overset{64}{>} 0.1 \overset{93}{>} \textit{0.01}$	croissant
setInitAlpha	$\mathbf{0.01} \overset{58}{>} 0.1 \overset{67}{>} 200 \overset{98}{\approx} \textit{500}$	constant
setInitD	$\mathbf{50} \overset{75}{>} 40 \overset{66}{>} 30 \overset{84}{>} \textit{20}$	croissant
Temps de paramétrage		11h

(c) iRFRLS

iRFRLS		
Paramètre	Classement	Temps de calcul
γ	$\mathbf{10^{-6}} \overset{99.9}{>} 10^{-12} \overset{96}{\approx} 10^{-3} \overset{100}{>} 1 \overset{100}{>} \textit{10}$	constant
λ	$10 \overset{99.9}{>} 1 \overset{99.9}{>} \textit{10}^{-3} \overset{97}{\approx} 10^{-6} \overset{96}{\approx} \mathbf{10^{-12}}$	constant
D	$50 \overset{100}{>} 500 \overset{99.9}{>} \mathbf{1000} \overset{99.9}{>} \textit{2000}$	$\mathcal{O}(D^2)$
Temps de paramétrage		12 jours



(a) (b) (c)

FIGURE 1: Étude expérimentale de LWPR, XCSF et iRFRLS. (a) Courbes d'erreur obtenues avec le meilleur et le pire jeu de paramètres pour LWPR, XCSF et iRFRLS. Remarquer l'échelle logarithmique. (b) Comparaison des meilleurs jeux de paramètres pour chaque algorithme. XCSF a été testé sur l'apprentissage des modèles géométrique et cinématique. (c) Comparaison de l'apprentissage en-ligne et hors-ligne. On représente la différence $\Delta MSE = MSE_{enligne} - MSE_{horsligne}$.

4.2 Comparaison des algorithmes d'apprentissage

Dans Sicard *et al.* (2011), les paramètres étaient optimisés à partir de données générées aléatoirement (pas de relation temporelle entre elles, excluant notamment toute notion de trajectoire). C'est pourquoi nous débutons par une approche similaire, avant de tester les algorithmes lorsque l'ordre de présentation des données est contraint par les trajectoires suivies par le robot.

4.2.1 Apprentissage hors-ligne

Dans un premier temps, les ensembles d'apprentissage et de test sont générés par tirage aléatoire des points de la base. La figure 1(b) compare les performances des trois algorithmes avec les meilleurs jeux de paramètres.

Nous remarquons tout d'abord que la performance finale d'XCSF est meilleure sur l'apprentissage de la jacobienne cinématique que sur l'apprentissage du modèle géométrique (confiance supérieure à 99.9%). Afin de comparer les meilleures performances de chacun des algorithmes, nous poursuivons l'étude d'XCSF sur l'apprentissage de la matrice jacobienne cinématique.

La convergence est quasiment instantanée pour LWPR et iRFRLS (les modèles sont correctement appris en moins de 500 itérations). En revanche, la convergence d'XCSF est beaucoup plus longue. Ceci confirme un résultat de Sicard *et al.* (2011) : XCSF converge plus lentement mais vers une erreur plus

faible que LWPR (confiance de 98.8%). Quant à lui, iRFRLS converge aussi rapidement que LWPR mais avec une performance finale meilleure (confiance supérieure à 99.9%). Mais, s'il converge plus rapidement qu'XCSF, la différence n'est significative qu'à 84%.

4.2.2 Apprentissage en-ligne

Nous utilisons maintenant des points consécutifs et non plus tirés aléatoirement. Nous mettons ainsi les algorithmes dans des conditions réelles d'utilisation, lorsque l'apprentissage est contraint par les mouvements du robot.

La figure 1(c) représente la différence entre l'erreur obtenue hors-ligne et celle obtenue en-ligne $\Delta MSE = MSE_{enligne} - MSE_{horsligne}$. Remarquons que si la convergence est plus lente, la performance finale est similaire pour iRFRLS et XCSF, alors qu'elle est meilleure pour LWPR. Cependant, le classement final des algorithmes est identique à celui de l'apprentissage hors-ligne : iRFRLS est toujours le plus rapide à converger et le meilleur en termes de performance finale.

4.3 Temps de calcul

Pour être utilisable sur des systèmes réels, les algorithmes doivent respecter des contraintes temporelles fortes, en particulier pour le temps de prédiction. La table 2 résume les temps de calcul des trois algorithmes [4].

TABLE 2: Temps de calcul moyen (maximal) en apprentissage et prédiction.

Algo \ Temps	Apprentissage	Prédiction
LWPR	50μs (1ms)	50μs (1ms)
XCSF	300 μs (20ms)	300μs (20ms)
iRFRLS, D=50	20μs (25μs)	20μs (25μs)
iRFRLS, D=2000	50ms (50ms)	450μs (500μs)

5 Discussion

Les résultats de la partie précédente plaident nettement en faveur d'iRFRLS. Cependant, certains points demandent une analyse approfondie.

4. Les expériences ont été faites sur un processeur Intel Core i7 960 avec 6Go de RAM.

5.1 Performance

Les résultats de la section 4.2 appellent deux discussions : l'une sur l'intérêt de l'optimisation, l'autre sur l'influence de l'ordre de présentation des données. La table 3 propose une synthèse de ces discussions.

5.1.1 Intérêt de l'optimisation

Si l'impact de l'optimisation est tangible avec les trois algorithmes (figure 1(a)), son intérêt est limité pour LWPR : ni la performance finale (gain inférieur à 2%) ni la vitesse de convergence ne sont impactées de manière significative. Il s'agit toutefois de l'optimisation la plus rapide des trois algorithmes (table 1(b)). L'optimisation est en revanche incontournable pour iRFRLS (gain d'un facteur 1000) ainsi que pour XCSF pour lequel les performances finales, la vitesse de convergence et la stabilité sont impactées.

5.1.2 Ordre de présentation des données

De manière évidente, la vitesse de convergence des algorithmes dépend du temps nécessaire pour couvrir suffisamment l'espace avec les données d'apprentissage. La convergence est donc plus lente lorsque la présentation des données suit l'ordre des trajectoires, par rapport à une présentation aléatoire. L'étude de l'impact sur les performances finales est plus pertinente.

La théorie garantit que la performance finale d'iRFRLS ne dépend pas de l'ordre de présentation des données, aux éventuelles erreurs d'arrondis près qui peuvent se cumuler lors du calcul incrémental de la factorisation de Cholesky. De même, l'algorithme génétique d'XCSF permet de réduire la dépendance à l'ordre de présentation des données. La figure 1(c) montre en effet une convergence rapide d'iRFRLS vers la même erreur que lors de l'apprentissage hors-ligne. XCSF est plus lent à converger, mais atteint le même niveau d'erreur.

Contrairement à XCSF, LWPR ne dispose pas d'un mécanisme d'optimisation de la position des RF. Puisque l'ordre de présentation des données détermine l'ordre de création des RF, et donc leur position, il modifie la performance finale (figure 1(c)). Cependant, contre toute attente, il faut souligner que la performance finale est meilleure lorsque les données sont présentées selon les trajectoires. Ceci peut s'expliquer par le fait qu'il est plus facile d'identifier et de filtrer correctement le bruit lorsque les données correspondent à des points consécutifs de trajectoires, en construisant des RF

TABLE 3: Synthèse de la comparaison de LWPR, XCSF et iRFRLS.

Algorithme	LWPR	XCSF	iRFRLS
Convergence	Rapide	Lent	Rapide et instable
Performance finale	-	+	++
Sensibilité au paramétrage	Faible	Grande	Extrême
Temps de calcul	Rapide	Variable, parfois lent	Compromis temps / précision
Avantages	Rapide et stable	Distinction condition / prédiction	Coût constant, peu de paramètres et bonnes performances
Inconvénients	Nombreux paramètres	Parfois instable et convergence lente, nombreux paramètres	Lent pour de grandes valeurs de D, sensibilité au paramétrage

qui se superposent le long de ces trajectoires et dont la combinaison donne une prédiction plus fiable. En revanche, dans le cas où le signal n'est pas aussi bruité, ce mécanisme peut jouer en défaveur de LWPR en lissant des variations propres à la fonction à apprendre. Ce point reste à étudier plus précisément.

5.2 Étude comparée

Nous analysons dans cette partie l'influence des paramètres au regard des spécificités de notre configuration expérimentale, dans le but de mettre en évidence quelques règles qui simplifient le paramétrage.

5.2.1 Bruit et régularité des fonctions à apprendre

Dans notre étude, la fonction à apprendre est très régulière avec des linéarités importantes. Ceci impacte le paramétrage des algorithmes et explique leur convergence rapide lors de l'apprentissage du modèle géométrique. À cause de la plus grande dimensionnalité du problème, l'apprentissage de la matrice jacobienne par XCSF est plus lent (figure 1(b)).

En revanche, apprendre le modèle géométrique pour calculer le modèle cinématique a pour effet d'augmenter la variance de l'erreur (puisque l'on se

sert deux fois du modèle pour calculer la vitesse), ce qui peut justifier la moins bonne performance finale d'XCSF avec le modèle géométrique.

Au delà de la régularité de la fonction à apprendre, les données utilisées sont très bruitées. Apprendre des modèles réguliers est d'autant plus important pour éviter le sur-apprentissage. Ceci favorise de large régions de validité pour les classeurs d'XCSF, ce qui se retrouve dans la valeur de *coverConditionRange* qui masque les valeurs de *minConditionStretch*. De même, une valeur plus grande pour ϵ_0 lisse les prédictions en prenant en compte plus de classeurs pour chaque prédiction. Pour éviter le sur-apprentissage, le paramétrage de l'algorithme génétique est crucial : il faut éviter une spécialisation de la population sur le bruit et favoriser un fort renouvellement. Ceci s'obtient à travers les paramètres *startCondensation* et Δ. En travaillant dans l'espace de Fourier, iRFRLS filtre le bruit plus efficacement, ce qui peut expliquer sa supériorité. Remarquons qu'un choix d'une valeur trop élevée pour γ, i.e. une sur-représentation des hautes fréquences, dégrade les performances.

Au contraire, LWPR semble favoriser des RF plus petits, à travers la grande valeur de *setInitD*. Cela peut sembler contradictoire avec l'analyse précédente, mais à l'inverse d'XCSF, LWPR n'optimise pas le centre de ces RF, et de plus petits RF permettent alors d'obtenir une meilleure précision et une plus grande flexibilité grâce à une partition plus fine de l'espace.

5.2.2 Complexité des modèles

Les tests de LWPR et XCSF mettent en évidence l'existence d'une taille de population minimale pour ces deux algorithmes. Pour XCSF, celle-ci se situe entre 100 et 500 classeurs, comme le montre la très grande différence de performances pour ces deux valeurs de *maxPopulationSize*, alors que le gain est beaucoup plus limité en passant de 500 à 1500 classeurs. Pour LWPR, l'effet se remarque sur *wGen*, seuil qui détermine la création de nouveaux RF. Le gain est très important entre 0.01 et 0.1, alors qu'il est beaucoup plus limité lors des autres augmentations.

L'étude du nombre d'estimateurs D d'iRFRLS est surprenante à première vue. En effet, ce nombre est directement relié à la qualité de l'estimation de la transformée de Fourier, et une forte valeur devrait toujours être préférable. Or l'étude met en évidence une performance moyenne décroissante avec le nombre d'estimateurs. À cause des fortes linéarités du modèle géométrique, un faible nombre de fréquences est suffisant (et préférable, si l'on veut éviter d'apprendre les discontinuités dues au bruit). De plus, le nombre

de coefficients à apprendre correspond au nombre d'estimateurs, et un faible nombre d'estimateurs facilite donc l'apprentissage. Ceci explique pourquoi en moyenne, avec des valeurs aléatoires pour les autres paramètres, il est plus facile d'obtenir de meilleures performances avec seulement 50 estimateurs, alors que 1000 estimateurs sont nécessaires pour avoir le meilleur modèle lorsque les autres paramètres sont correctement choisis.

5.2.3 Temps de calcul

Le temps de calcul est une caractéristique importante des algorithmes d'apprentissage qui rend possible ou non leur utilisation pour des applications temps-réel. La table 2 compare les temps d'apprentissage et de prédiction pour chacun des trois algorithmes.

LWPR est le plus rapide des trois, ce qui le rend facilement utilisable en temps-réel. L'influence des paramètres (table 1(b)) n'est donc pas déterminante.

Pour XCSF, le temps de calcul dépend principalement de la taille de la population et du moment où la condensation est activée (table 1(a)). Cependant, ce dernier effet ne se ressent que lors de la phase d'apprentissage : une fois que la population est stable après la condensation, le temps de prédiction devient constant. En moyenne, les temps de calcul sont d'environ 300μs, avec un maximum de 20ms généralement atteint juste avant la condensation.

Le temps de calcul d'iRFRLS ne dépend que d'un seul paramètre : le nombre d'estimateurs (table 1(c)). Ceci permet de rechercher facilement un compromis temps de calcul / précision. De 20μs pour $D = 50$, on atteint 50ms en apprentissage et 450μs en prédiction avec $D = 2000$. De tels temps de calcul peuvent nécessiter un apprentissage hors-ligne ou d'importantes ressources de calcul pour une utilisation temps-réel.

5.3 Règles de paramétrage

À travers cette discussion, nous avons identifié quelques règles qui facilitent le paramétrage des algorithmes en fonction du problème considéré. Elles sont présentées table 4. La table doit être lue de la manière suivante : en fonction de la régularité de la fonction à apprendre, les principaux paramètres de LWPR à modifier sont ceux contrôlant la taille des RF et leur superposition. Pour éviter le sur-apprentissage avec LWPR, il faut modifier les paramètres contrôlant l'optimisation des RF et le méta-apprentissage, etc.

TABLE 4: L'identification de quelques propriétés et de leurs paramètres associés permet de simplifier le paramétrage.

Algo.	Régularité de la fonction	Sur-apprentissage	Complexité
LWPR	Taille et superposition des RF	Optimisation des RF et meta-apprentissage	Toujours rapide
XCSF	Taille et nombre de classeurs	Condensation et algorithme génétique	Taille de la population
iRFRLS	Échantillonnage des pulsations		Nombre d'estimateurs

6 Conclusion et perspectives

Dans ce travail, nous avons appris des modèles du robot iCub et montré que LWPR, XCSF et iRFRLS avaient de bonnes performances en présence d'importantes sources de bruit. iRFRLS surpasse XCSF et LWPR, en termes de vitesse de convergence et de performances. Cependant, le temps de calcul d'iRFRLS peut devenir prohibitif pour des applications temps-réel. Il reste toutefois possible d'assurer un fonctionnement temps-réel à la conception, grâce à un compromis temps de calcul / précision, à travers un paramétrage pertinent du nombre d'estimateurs. Nous avons également mis en évidence que l'optimisation des paramètres est cruciale pour iRFRLS et XCSF, alors que le gain en performances est beaucoup plus faible pour LWPR. De plus, si l'apprentissage en-ligne est plus lent, ce qui s'explique principalement par le temps nécessaire pour couvrir tout l'espace, l'impact sur les performances finales est réduit pour les trois algorithmes.

De futures études consisteront à valider nos résultats en ligne sur le robot. Ceci apportera deux améliorations à cette étude : il sera possible de tester la performances des algorithmes sur de plus grands espaces, avec de moins grandes linéarités dans les fonctions à estimer, et les performances pourront être évaluées non seulement sur l'erreur quadratique moyenne des prédictions, mais aussi sur la qualité des trajectoires obtenues. À plus long terme, nous désirons également étudier l'influence de l'introduction de mécanismes de curiosité artificielle sur les algorithmes.

Remerciements

Ce travail est soutenu par le programme français ANR (ANR 2010 BLANC 0216 01), plus d'informations sur http ://macsi.isir.upmc.fr

Références

BJÖRCK Å. (1996). *Numerical Methods for Least Squares Problems*. Philadelphia : SIAM.

BUTZ M. V. & HERBORT O. (2008). Context-dependent predictions and cognitive arm control with XCSF. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, GECCO '08, p. 1357–1364, New York, NY, USA : ACM.

DEO A. & WALKER I. (1995). Overview of damped least-squares methods for inverse kinematics of robot manipulators. *Journal of Intelligent and Robotic Systems*, **14**, 43–68.

DRUCKER H., KAUFMAN C., BURGES L., SMOLA A. & VAPNIK V. (1997). Support vector regression machines. In *Advances in Neural Information Processing Systems 9*, volume 9, p. 155–161.

GIJSBERTS A. & METTA G. (2011). Incremental learning of robot dynamics using random features. In *ICRA*, p. 951–956.

HOLLAND J. H. (1992). *Adaptation in Natural and Artificial Systems : An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press.

KAILATH T., SAYED A. & HASSIBI B. (2000). *Linear estimation*. Prentice-Hall information and system sciences series. Prentice Hall.

METTA G., SANDINI G., VERNON D., NATALE L. & NORI F. (2008). The icub humanoid robot : an open platform for research in embodied cognition. In *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, p. 50–56.

PATTACINI U., NORI F., NATALE L., METTA G. & SANDINI G. (2010). An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots. In *IROS*, p. 1668–1674.

RAHIMI A. & RECHT B. (2008). Random features for large-scale kernel machines. In J. PLATT, D. KOLLER, Y. SINGER & S. ROWEIS, Eds., *Advances in Neural Information Processing Systems 20*, p. 1177–1184. Cambridge, MA : MIT Press.

RUXTON G. D. (2006). The unequal variance T-test is an underused alternative to Student's T-test and the mann-whitney U test. *Behavioral Ecology*, **17**(4), 688–690.

SALAUN C., PADOIS V. & SIGAUD O. (2009). Control of redundant robots using learned models : an operational space control approach. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, p. 878–885, Saint-Louis, USA.

SAYED A. H. (2008). *Adaptive Filters*. Wiley-IEEE Press.

SCHAAL S., ATKESON C. G. & VIJAYAKUMAR S. (2002). Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence*, **17**, 49–60.

SICARD G., SALAÜN C., IVALDI S., PADOIS V. & SIGAUD O. (2011). Learning the velocity kinematics of iCub for model-based control : XCSF versus LWPR. In *Proceedings of the 11th IEEE-RAS International Conference on Humanoid Robots - Humanoids 2011*.

SIGAUD O., SALAÜN C. & PADOIS V. (2011). On-line regression algorithms for learning mechanical models of robots : a survey. *Robotics and Autonomous Systems*, **59**, 1115–1129.

STALPH P. O. & BUTZ M. V. (2010). Current XCSF capabilities and challenges. In *IWLCS 2008/2009*, p. 57–69 : Springer.

VIJAYAKUMAR S. & SCHAAL S. (2000). Locally weighted projection regression : An o(n) algorithm for incremental real time learning in high dimensional space. In *in Proceedings of the Seventeenth International Conference on Machine Learning 2000*, p. 1079–1086.

WILSON S. W. (1995). Classifier Fitness Based on Accuracy. *Evolutionary Computation*, **3**(2), 149–175.

WILSON S. W. (2001). Function approximation with a classifier system. *Genetic and Evolutionary Computation Conference, GECCO 2001*, p. 974–981.

WOLD H. (1975). Soft Modeling by Latent Variables ; the Nonlinear Iterative Partial Least Squares Approach. *Perspectives in Probability and Statistics. Papers in Honour of M. S. Bartlett*.