Metode neparametrice în instruirea automată a maşinilor: aplicaţii în robotică şi analiza datelor

Raport ştiinţific

Lehel Csató

10 octombrie 2014

Rezumat

Documentul prezintă rezultatele ştiinţifice obţinute în perioada 2011–2014 în cadrul proiectului **PN-II-RU-TE-2011-3-0278** dedicat grupurilor de cercetare cu cercetători tineri. În document sunt detaliate rezultatele ştiinţifice obţinute în cadrul proiectului, defalcate pe sub-categorii, conform cu tabelul de mai jos.

Adresa de web a proiectului este: http://datamin.ubbcluj.ro/index.php?a=projects;
Tematica seminariilor - defalcată pe categorii – organizate cu membrii echipei de cercetare respectiv pentru atragerea de noi membri este la pagina: http://datamin.ubbcluj.ro/index.php?a=semi#semi.

Cuprins

1 Introducere

Domeniul de cercetare a metodelor neparametrice şi a aplicării acestora în instruirea automată cuprinde mai multe subdomenii importante, toate aceste subdomenii au ca caracteristică comună potenţialul ca mulţimea de date de intrare să fie mare. Proiectul se ocupă de studiul metodelor de instruire automată a maşinilor, axându-se pe metode de inferenţă neparametrice şi pe metodologia Bayes, cu aplicaţii în robotică respectiv în procesul de analiză a datelor, cu accent pe analiza datelor textuale. Această caracteristică justifică eforturile de realizare a algoritmilor ce pot prelucra aceste date în mod *automat* – ca şi în metodele de instruire prin întăriri – sau cu intervenţii umane minimale – de exemplu în metodele semi-supervizate. Avantajul metodelor neparametrice este ajustarea "automată" – în funcţie de caracteristicile datelor de instruire – a complexităţii rezultatelor, fapt realizat prin adăugarea sau ştergerea eficientă a unor parametri. Direcţiile de cercetare pe care ne-am concentrat de-a lungul proiectului au fost:

1. Aplicarea proceselor Gaussiene – metode neparametrice probabiliste – în developarea de algoritmi pentru instruirea prin întăriri, mai exact cu algoritmi aproximativi de instruire prin întăriri (approximate reinforcement learning).
2. Aplicarea proceselor Gaussiene în realizarea de algoritmi noi pentru robotică.
3. Aplicarea metodelor neparametrice pentru îmbunătăţirea algoritmilor cu date de intrare imprecise.
4. Aplicarea metodelor neparametrice pentru algoritmi de căutare rapidă respectiv pentru realizarea de algoritmi semi-supervizaţi.

2 Algoritmi aproximativi de instruire automată prin întăriri

Un subdomeniu de aplicare cu succes a metodelor neparametrice este acea a instruirii prin întăriri aproximative – *approximate reinforcement learning(ARL)*. Importanţa acestor cercetări este că – contrar metodelor clasice de instruire – ARL poate fi aplicat cu uşurinţă la probleme de control şi de învăţare robotică din lumea reală, fără modelarea dinamicii sau a cinematicii roboţilor. Caracteristicile principale ale acestor probleme sunt:

- Spaţii de stări şi acţiuni continue şi de dimensionalitate ridicată;
- Necesitatea de rulare a algoritmilor *on-line*;
- Diversitate problemelor la nivel de control, ceea ce duce la dificultatea construirii aproximatoarelor de funcţii potrivite pentru fiecare domeniu de aplicaţie;
- Numărul limitat de încercări – *trial* – care pot fi executate pentru a facilita învăţarea;
- Necesitate reducerii costurilor computaţionale pentru a facilita aplicarea metodelor pe unităţi mobile.

Ţinând cont de caracteristicile menţionate, putem identifica direcţii majore care constituie baza cercetării din cadrul proiectului. Asocierea unor valori de utilitate cu anumite stări sau perechi de stări şi acţiuni stă la baza metodelor clasice de instruire prin întăriri. Îndată ce spaţiul de stări sau stări-acţiuni devine infinit, trebuie să recurgem la aproximatoare de funcţii pentru realizarea acestui lucru. Metodele clasice de aproximare de funcţii trebuie adaptate la fiecare domeniu de aplicare pentru a putea fi utilizate eficient. Metodele neparametrice pe de altă parte au posibilitatea de a adapta deoarece numărul parametrilor nu este fixată ci este determinată de datele de instruire şi caracteristicile acestora. Utilizarea proceselor Gaussiene pentru acest scop şi studierea caracteristicilor acestora este una dintre temele principale a cercetării noastre.

Procesele Gaussiene pot fi folosite *on-line* ceea ce rezultă în metode folosibile pentru a acumula informaţii în acelaşi timp cu interacţiunea cu mediul înconjurător. Însă acesta dă naştere anumitor probleme precum: creşterea nelimitată a numerelor de parametri, schimbarea caracteristicilor funcţiilor aproximate în timpul procesului de învăţare şi apariţia relaţiilor temporale între măsurătorile făcute la fiecare pas a experimentelor. În cadrul proiectului am studiat unele posibilităţi de reducere a costurilor computaţionale prin eliminarea unor măsurători irelevante şi prin asocierea unor factori de importanţă cu fiecare *experienţă* făcută în timpul învăţării.

2.1 Îmbunătăţirea aproximării funcţiilor de valori

În anul calendaristic 2011 ne-am ocupat cu aproximarea valorilor stărilor şi a perechilor de stări-acţiuni, care este o metodă de bază în studiul problemelor de instruire prin întăriri. Cadrul teoretic generic este aceea a proceselor de decizii Markov – în engleză *"Markov Decision Processes"* – MDP [Puterman, 1994], definit printr-un cuartet $M\,(S, A, P, R)$, unde S este spaţiul stărilor, A spaţiul acţiunilor posibile, $P(s'|a, s)$ descrie probabilitatea condiţională ca starea s' să fie selectată în urma aplicării acţiunii a când suntem în starea s, iar $R(s)$ este funcţia de câştig – reward – asociat fiecărei stări s a sistemului. Rezolvarea unui MDP constă în calculul unei *strategii*

optime, aceasta fiind definită ca $\pi(a|s)$ şi este interpretată ca probabilitatea condiţională a selecţiei unei acţiuni a în starea s. Determinarea strategiei optime se face prin funcţiile de valoare $V(s)$ respectiv $Q(s,a)$ [Sutton and Barto, 1998]. O funcţie de valoare poate fi definită ca $V_\pi : S \to \mathbb{R}$, respectiv $Q_\pi : S \times A \to \mathbb{R}$:

$$V_\pi(s) = E_\pi\left[\sum_{t=0}^{\infty} \gamma^t R_t | s_0 = s\right], \qquad Q_\pi(s,a) = E_\pi\left[\sum_{t=0}^{\infty} \gamma^t R_t | s_0 = s, a_0 = a\right]$$

În formulele de mai sus $s \in S$ reprezintă o stare, $a \in A$ reprezintă o acţiune şi $\pi : S \times A \to [0,1]$ reprezintă o strategie pe baza căruia se alege o acţiune într-o anumită stare. Aplicarea metodelor RL la probleme din lumea reală necesită introducerea unor aproximatori de funcţii pentru a modela atât funcţiile de valori V şi Q cât şi modelarea strategiei $\pi(s,a)$. Folosirea proceselor Gaussiene pentru aproximarea funcţiilor de valori a fost cercetată în numeroase articole [Rasmussen and Williams, 2006, Ghavamzadeh and Engel, 2007, Deisenroth et al., 2009].

Una dintre problemele tratate de noi în [Jakab and Csató, 2011] este acea a continuităţii şi a refolosirii datelor de antrenament pentru aproximarea funcţiilor de valori. Ideea de bază este de a folosi o versiune modificată a metodei de sparsificare bazată pe distanţa Kullback-Leibler între distribuţiile posterioare a două procese Gaussiene.

2.2 Instruire prin întăriri cu căutare direcţionată a strategiei şi procese Gausiene

Aproximarea probabilistică a funcţiilor de valori poate fi exploatată şi în cazul metodelor de tip *policy gradient* (PG) cu scopul de a micşora varianţa gradientului estimat şi de a influenţa direcţiile de explorare în timpul procesului de instruire. Inferenţa – găsirea – unui proces Gaussian pentru a aproxima o funcţie de valoare poate fi realizată în felul următor: ca şi date input de instruire folosim perechi de stări-acţiuni $x_t \stackrel{\text{def}}{=} (s_t, a_t)$, iar outputurile corespunzătoare sunt valorile de utilitate "discontate": $\sum_{i=0}^{H-t} \gamma^i R(s_{t+i}, a_{t+i})$. Pentru antrenarea proceselor Gaussiene on-line folosim metodele dezvoltate în [Csató and Opper, 2002].

2.2.1 Influenţarea direcţiei explorării în problemele de tip "reinforcement learning"

Metodele de tip PG necesită o reprezentare explicită a strategiei de luare de decizii $\pi_\theta(s,a)$ în forma unei aproximatori de funcţii parametrice. În [Jakab and Csató, 2012] am introdus două posibilităţi de îmbunătăţire a explorării. Prima se referă la influenţarea magnitudinii zgomotului adăugat la o strategie având forma:

$$\begin{aligned} \pi_\theta &= c(s,\theta_c) + \mathcal{N}(0, \sigma_{GP}^2 I) \\ \sigma_{GP}^2 &= \lambda\left(k_q(x^*,x^*) - \mathbf{k}^* \mathbf{C}_n \mathbf{k}^{*T}\right), \quad \text{având } x^* = \{s, c(s,\theta_c)\} \end{aligned} \tag{1}$$

În ecuaţia de mai sus $\mathbf{k}^* = [k_q(x^*,x_1), \ldots, k_q(x^*,x_n)]$ este un vector compus din valorile covarianţelor punctului x^* cu restul datelor din setul de instruire $\mathcal{D}$, σ_{GP}^2 este varianţa procesului Gaussian folosit pentru aproximarea funcţiei de valori, evaluat în punctul x^* compus dintr-o stare s şi acţiunea selectată de către controlor în această stare, $c(s,\theta_c)$ – funcţie care este dată de funcţia medie a procesului Gaussian.

Pentru a influenţa atât magnitudinea varianţei cât şi a direcţiei explorării, în [Jakab and Csató, 2012] am introdus o strategie care foloseşte valoarea medie a procesului Gaussian, evaluat într-un număr de puncte din proximitatea perechii de stare-acţiune în care se află sistemul:

$$\pi(a|s) = \frac{e^{\beta E(s,a)}}{Z(\beta)}, \text{ unde } Z(\beta) = \int da\, e^{\beta E(s,a)} \text{ este constanta de normalizare a probabilităţii.} \tag{2}$$

Formularea de mai sus permite o strategie stohastică, a cărei fluctuaţie poate fi controlat de valoarea β, fiind inversa "temperaturii sistemului". Alegerea acţiunii a "optime" din starea s, prin strategia $\pi(a|s)$, este făcută cu ajutorul prin formularea unei funcţii de energie $E(s,a)$ care, lângă funcţia valoare $Q(s,a)$, are şi un factor multiplicativ de penalizare a acţiunilor la distanţă mare de cea selectată $c_\theta(s)$:

$$E(s,a) \quad = \quad \hat{Q}_{GP}(s,a) \exp\left[-\frac{\| a - c_\theta(s) \|^2}{2\sigma_e^2}\right]$$

2.2.2 Reducerea varianţei gradientului

O altă dificultate pe care am încercat să soluţionăm este problema varianţei în valoarea gradientului [Jakab and Csató, 2012]. În metodele "policy gradient", în fiecare pas al procesului de instruire se calculează gradientul

funcţiei – *objective function* – $J^{\pi} = E_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1}\right]$ relativ la parametrii funcţiei $\pi(a|s,\theta)$, θ. Gradientului se calculată folosind aproximări de tip Monte Carlo, având forma:

$$\nabla_{\theta} J = E_{\tau}\left[\sum_{t=0}^{H-1} \nabla_{\theta} \log \pi(a_t|s_t) \sum_{i=0}^{H-t} \gamma^i R(s_{t+i}, a_{t+i})\right] \tag{3}$$

unde τ reprezintă o serie de înregistrări – en "rollout" – rezultate prin rularea controlorului având lungimea H, iar $E[\cdot]_{\tau}$ este valoarea medie relativă la distribuţia "rollout-ilor" τ. Principala slăbiciune a metodelor PG este varianţa mare a valorii gradientului, ca rezultat a estimării prin metoda Monte Carlo: $\sum_{i=0}^{H-t} \gamma^i R(s_{t+i}, a_{t+i})$. În [Jakab and Csató, 2012] am arătat că, prin folosirea unui proces Gaussian pentru aproximarea funcţiei de valori, respectiv prin înlocuirea mediei empirice cu valoarea medie procesului Gaussian posterior, varianţa gradientului poate fi redusă în mod semnificativ:

$$\nabla_{\theta} J(\theta) = E_{\tau}\left[\sum_{t=0}^{H-1} \nabla_{\theta} \log \pi(a_t|s_t) \hat{Q}_{GP}(s_t, a_t)\right], \tag{4}$$

unde $\hat{Q}_{GP}(\cdot,\cdot)$ este funcţia de valoare aproximată cu ajutorul unui proces Gaussian, iar articolul conţine şi analiza empirică a performanţei acestei modificări a fost elaborată în articolul [Jakab and Csató, 2012].

2.3 Aproximări neparametrice bazate pe manifold-uri

Una dintre problemele aproximării funcţiilor de valori şi a strategiilor cu ajutorul proceselor Gaussiene reprezintă apariţia discontinuităţilor în suprafeţele funcţiilor aproximate. Pentru a reprezenta mai exact discontinuităţile, am introdus o familie de funcţii kernel care profită de relaţia temporala între punctele de antrenament:

$$k_{sp}(x, x') = A \exp\left(-\frac{\mathcal{SP}(x, x')^2}{2\sigma_{sp}}\right) \tag{5}$$

unde A şi σ_{sp} sunt hyper-parametrii sistemului. Distanţa cea mai scurtă dintre punctele x şi x' poate fi calculată cu ajutorul conexiunilor unui graf construită on-line:

$$E_{x_t, x_i} = \begin{cases} \|x_i - x_t\|^2 & \text{dacă } \|x_i - x_t\| < \varepsilon \quad \varepsilon > 0 \\ 0 & \text{altfel} \end{cases} \tag{6}$$

Valoarea de prag ε limitează numărul vecinilor punctului x_t. Am folosit notaţia E_{x_t,x_i} pentru a nota o lamă din graful $G(V, E)$ întru nodurile x_t şi x_i Cu informaţia extrasă din relaţia temporară construim o măsură de distanţă nouă care reflectă distanţa cea mai scurtă dea-lungul suprafeţei pe care se află punctele de antrenament întâlnite. Deoarece în majoritatea problemelor punctele de antrenament sunt continue, folosim două metode pentru a calcula distanţa între x^* şi punctul deja incorporat x_j:

$$\begin{aligned} \mathcal{SP}(x^*, x_j) &\overset{(1)}{=} \|x^* - x_i\|^2 + \mathbf{P}_{i,j}, \text{ cu } x_i = \operatorname*{argmin}_{x_\ell \in BV} \|x^* - x_\ell\|^2 \\ \mathcal{SP}(x^*, x_j) &\overset{(2)}{=} \mathbf{k}_{x*}^T \mathbf{P} \mathbf{e}_j = \sum_{i=1}^{n} k(x^*, x_i) \mathbf{P}_{i,j} \end{aligned}$$

unde matricea $\mathbf{P}$ conţine distanţa cea mai scurtă între x_i şi x_j, iar $\mathbf{e}_j$ este al j-lea vector de unitate cu lungime n. Detaliile acestei metode, precum şi evaluarea performanţei cu ajutorul unor probleme de tip reinforcement learning simulate, au fost publicate în articolul [Jakab and Csató, 2012].

2.4 Modele inteligente pentru modelarea comportamentului roboţilor

În martie 2012 a avut loc susţinerea tezei de doctorat al lui Jakab Hunor [Jakab, 2012] întitulată *Intelligent Models for Robotic Behavior, Decision Making and Environment Interaction*. Teza conţine detaliile cercetării efectuate în perioada 2009-2012 din care o parte semnificativă a fost realizată în timp ce Jakab Hunor era implicat în proiectul de faţă. Teza conţine 7 capitole şi se ocupă de modalităţi de îmbunătăţire a metodelor de instruire prin întăriri cu ajutorul metodelor neparametrice, în contextul aplicării lor la probleme de control din lumea reală.

2.5 Metode de sparsificare a rezultatelor algoritmilor de tip "reinforcement learning"

In anul calendaristic 2013 am studiat metodele de sparsificare a rezultatelor algoritmilor neparametrici. Pentru aceasta am considerat o metodă alternativă la aproximarea funcţiilor de valori, numită *Least squares value approximation* (LSVA) introdus în [Bradtke et al., 1996]. Principiul de bază a acestei metode este instruirea pe bază de diferenţe temporale – engleză *temporal difference learning* (TD) – cunoscută din literatura clasică de instruire prin întăriri [Sutton and Barto, 1998]. Adaptarea acestui algoritm la problemele de dimenzionalitate mare a fost realizată prin "kernelizarea" lor, ajungând astfel la forma care stă la baza metodelor introduse de noi:

$$\tilde{V}(s) = \sum_{i=0}^{n} \mathbf{w}_i^* k(s_i, s_t) \quad s \in S, \tag{7}$$

iar (re)calcularea $\mathbf{w}_i^*$ este realizată prin actualizarea iterativă a unor structuri de date pe baza ecuaţiilor:

$$\mathbf{w}^* = \tilde{A}^{-1}\tilde{b}, \text{ where } \tilde{A} = \frac{1}{n}\sum_{t=0}^{n} \mathbf{k}(s_t)\left[\mathbf{k}^T(s_t) - \gamma \mathbf{k}^T(s_{t+1})\right], \quad \tilde{b} = \frac{1}{n}\sum_{t=0}^{n} \mathbf{k}(s_t) R_t. \tag{8}$$

Aici, ca şi în secţiile anterioare, $k(\cdot,\cdot)$ este o funcţie kernel, iar $\mathbf{k}(s) = [k(s, s_1), \ldots, k(s, s_n)]^T$ este vectorul de valori ale funcţiei kernel evaluate pe punctul de date s şi datele de antrenament $s_i \quad i = \overline{1, n}$. Forma aceasta a actualizărilor iterative permite tratarea metodei prezentate ca o aproximare neparametrică: în fiecare moment când primim un nou punct de instruire (prin măsurători de exemplu), putem decide dacă vrem să păstrăm acel punct şi a-l încorpora în structurile de date. Acest proces se numeşte *sparsificare* şi este necesară deoarece costul calculării inversei matricei $\tilde{A}$ este cubic [Csató and Opper, 2002].

2.5.1 Sparsificarea datelor

În spiritul celor de sus, în 2013 ne-am ocupat de problema *dependenţei lineare aproximative – Approximate linear dependency* –, o metodă frecvent întâlnită pentru realizarea sparsificării. Această metodă păstrează punctele de date care sunt greu expresibile prin combinarea lineară a punctelor deja existente. În [Jakab and Csató, 2013] am introdus o metodă de sparsificare bazată pe caracteristicile unui graf construit on-line între punctele de antrenament. Această metodă foloseşte operatorul Laplacian pentru formularea unei criterii $\mu(\cdot)$ pe baza căruia se decide incorporarea unui ponct de instruire: importanţa unui vertex $s_i \in \mathcal{V}$ este exprimată ca suma ponderată a diferenţei pătrate între valoarea funcţiei de ţintă a punctelor s_i şi vecinilor lui: $\mu(s_i \in \mathcal{V}) = \sum_{v_j \in \text{neig}(s_i)} A_{ij}\left[f(s_i) - f(s_j)\right]^2$. Calitatea mulţimii $\mathcal{V}$ se exprimă cu acelaşi operator Laplacian:

$$\sum_{s_i \in \mathcal{V}} \mu(s_i) = \sum_{i,j=1}^{n} A_{ij}\left[f(s_i) - f(s_j)\right]^2 = \mathbf{f^T L f}. \tag{9}$$

Pentru construirea grafului $G(\mathcal{E}, \mathcal{V}), \quad \mathcal{V} \subset S$ am studiat două metode diferite: construcţie bazată pe *k nearest neighbour* (KNN) şi construcţia bazată pe *Extended sphere of influence* (eSIG). Detaliile algoritmilor de construcţie şi actualizare a acestora sunt detaliate în articolul [Jakab and Csató, 2013]. Rezultatele experimentale arată că, folosind metoda de sparsificare ajută la distribuirea mulţimii $\mathcal{V}$ în regiunile importante ale spaţiului de căutare, având ca rezultat funcţii mai stabile şi mai exacte decât cele bazate pe metodele clasice.

2.5.2 Îmbunătăţiri aduse algoritmilor de sparsificare

În anul calendaristic 2014 – publicat în articolul [Jakab and Csató, 2014] – am îmbunătăţit metodele LSVA "sparse". Una dintre rezultatele importante este dezvoltarea regulilor de actualizare on-line a structurilor de date $\tilde{A}$ şi $\tilde{b}$ din ecuaţia (8) prin care devine posibilă micşorarea costului computaţional şi de memorie al algoritmului. Principiul de bază a metodei este de a calcula incremental inversul matricei $C = \tilde{A}$ fără a păstra matricea originală. Ecuaţiile de actualizare capătă forma următoare:

$$C_{t+1} = \tilde{A}_{t+1}^{-1} = \frac{t+1}{t}\begin{bmatrix} \tilde{A}_t + \mathbf{u}\mathbf{v}^T & v^*\mathbf{u} \\ u^*\mathbf{v}^T & u^*v^* \end{bmatrix}^{-1} = \begin{bmatrix} C_t & -C_t\mathbf{u}/u^* \\ -\mathbf{v}^T C_t/v^* & \frac{1+\mathbf{v}^T C_t \mathbf{u}}{u^* v^*} \end{bmatrix} \tag{10}$$

Pentru a obţine o expresie mai compactă am folosit notaţiile:

$$\mathbf{u} = [k(s_t, s_1), \ldots, k(s_t, s_t)]^T \qquad u^* = k(s_t, s_{t+1})$$
$$\mathbf{v} = [k(s_t, s_1) - \gamma k(s_{t+1}, s_1), \ldots, k(s_t, s_t) - \gamma k(s_{t+1}, s_t)]^T \qquad v^* = k(s_t, s_{t+1} - \gamma k(s_{t+1}, s_{t+1})$$

Pentru obţinerea formei analitice am folosit regulile de inversie a matricelor bloc şi formula Sherman-Woodbury, ca şi în [Csató and Opper, 2002]. Expresia finală pentru obţinerea valorii coeficienţilor de aproximare devine surprinzător de simplă:

$$\begin{aligned}\mathbf{w}_{t+1} &= C_{t+1}\tilde{\mathbf{b}}_{t+1} = \begin{bmatrix} C_t & -C_t\mathbf{u}/u^* \\ -\mathbf{v}^T C_t/v^* & \frac{1+\mathbf{v}^T C_t \mathbf{u}}{u^* v^*} \end{bmatrix} \left(\begin{bmatrix} \tilde{\mathbf{b}}_t \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{u} \\ u^* \end{bmatrix} R_t \right) \\ &= \begin{bmatrix} \mathbf{w}_t \\ \left(R_t - \mathbf{v}^T \mathbf{w}_t\right)/v^* \end{bmatrix} = \begin{bmatrix} \mathbf{w}_t \\ \left(R_t - (\tilde{V}(s_t) - \gamma \tilde{V}(s_{t+1}))\right)/v^* \end{bmatrix} \end{aligned} \tag{11}$$

unde forma actualizată a $\tilde{\mathbf{b}}_t$ este: $\tilde{\mathbf{b}}_{t+1} = \frac{t}{t+1}\left(\begin{bmatrix} \tilde{\mathbf{b}}_t \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{u} \\ u^* \end{bmatrix} R_t \right)$ Posibilitatea de a executa calculele necesare on-line contribuie semnificativ la aplicabilitatea metodelor prezentate la probleme reale unde datele de antrenament sunt obţinute secvenţial prin interacţiune cu mediul în care se desfăşoară procesul de instruire. Totodată datorită actualizărilor secvenţiale costurile computaţionale sunt reduse de la $O(n^3)$ la $O(nk)$ unde n este numărul punctelor de antrenament considerate semnificativ din punt de vedere a cantităţii de informaţie conţinută.

De asemenea în articolul [Bócsi et al., 2014] am investigat noi posibilităţi de construire a grafurilor de proximitate pentru a îmbunătăţi metodele de sparsificare relatate mai sus.

3 Învăţarea modelelor de roboţi pentru control de urmărire

În general, controlul roboţilor este bazat pe un modelul matematic al robotului. În mod tradiţional acest model este definit prin parametri fizici ai robotului. Bazat pe acestea, configuraţia de articulaţie dorită – numit model cinematic, sau forţele dorite – numit modelul dinamic, au forme analitice. Această abordare este folosită în cazuri când arhitectura robotului este fixă şi condiţiile externe nu se schimbă. În condiţiile în care arhitectura robotului şi mediul de execuţie sunt fixe, modelele analitice rezultă în control precis şi eficient. În cazurile când condiţiile externe sunt diferite sau modelul robotului nu este fixă, soluţiile analitice au dezavantaje serioase: ori modelul matematic este inadecvat robotului ori este prea complex pentru a putea fi modelat eficient. Am propus o abordare adaptivă pentru a obţine modelul robotului. Scopul nu este de a defini un model de control bazat pe structura fizică a robotului ci ca o funcţie dintre datele de intrare senzoriale şi datele de ieşire de control, obţinând o *aproximare eficientă a modelului*. În cele ce urmează, definim problema de control de urmărire.

3.1 Metode de control pentru urmărire

De obicei, problema de control de urmărire – en "tracking control – este formulată în spaţiul de efector – en "task-space" – când efectorul robotului trebuie să-şi urmeze o traiectorie predefinită. Soluţia problemei anterioare nu este unică. Pentru roboţi redundanţi, funcţia din spaţiul de efector la spaţiul de articulaţii (joint space) nu este unică: pentru o poziţie de efector există mai multe configuraţii de articulaţii care formează un spaţiu neconvex de soluţii . În [Bócsi et al., 2011b,a, 2012, 2014a] am propus un algoritm bazat pe învăţarea automată care este capabil de a şi controla roboţi nerigizi unde soluţiile standarde nu funcţionează.

Algoritmul de control de urmărire are trei paşi: (1) aproximăm un model comun al coordinatelor de efector şi al coordinatelor de articulaţii folosind tehnici de învăţare automată. (2) aplicăm optimizare locală pentru a obţine cinematica inversă, notată cu f^{-1}; (3) bazat pe cinematica inversă, folosim un controlor de articulaţii la toate gradele de libertate – en "degrees of freedom" DoF– ale robotului pentru a obţine forţele necesare.

3.2 Învăţarea indirectă a modelelor în robotică

Observaţia principală este că un model $E(\mathbf{x}, \theta)$ dintre datele de intrare şi ieşire este bine definit, iar predicţii pot fi obţinute prin minimizarea erorii modelului la un punct de intrare dat, *i.e.*,

$$f^{-1}(\boldsymbol{x}) \quad \stackrel{\circ}{=} \quad \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\operatorname{argmin}}\, E(\boldsymbol{x}, \boldsymbol{\theta}), \tag{12}$$

unde $\mathbf{x}$ este poziţia de efector şi θ este poziţia în spaţiul de articulaţii. O întrebare importantă este minimizarea din ecuaţia (12), respectiv şi problema de ne-unicitate a funcţiei de cinematică inversă: cum să efectuăm minimizarea în cazul în care o poziţie de efector $\boldsymbol{x}^{\text{desired}}$ poate fi atinsă de mai multe articulaţii $\boldsymbol{\theta}_1$ şi $\boldsymbol{\theta}_2$ (Fig. 1)? În acest

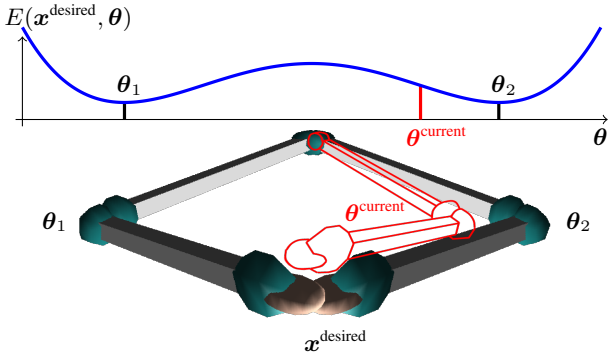


Figura 1: Ilutaţia schemei de predicţie a funcţiei de cinematică inversă. Poziţia $\boldsymbol{x}^{\text{desired}}$ se poate obţine din două configuraţii de articulaţie $\boldsymbol{\theta}_1$ şi $\boldsymbol{\theta}_2$, deci $E(\boldsymbol{x}^{\text{desired}}, \boldsymbol{\theta}_1) = E(\boldsymbol{x}^{\text{desired}}, \boldsymbol{\theta}_2)$. Algoritmul va alege $\boldsymbol{\theta}_2$, această configuraţie fiind mai aproapa de cea curentă $\boldsymbol{\theta}^{\text{current}}$.

caz, algoritmul trebuie să dea ca predicţie $\boldsymbol{\theta}_2$ pentru a evita mişcări bruşte. Acest comportament este favorabil pentru a obţine traiectorii netede de articulaţii. Am propos de a începe căutarea de gradient de la poziţia curentă de articulaţie $\boldsymbol{\theta}^{\text{current}}$ a robotului.

A doua întrebare importantă este cel legat de modelul $E(\cdot, \cdot)$ pentru a obţine un algoritm eficient, folosibil în aplicaţii din lumea reală. Am propus trei abordări posibile, folosind (1) *joint kernel support estimation*, (2) *structured output Gaussian processes*, şi (3) o metodă bazată pe cinematica directă a robotului. Folosind metoda **"joint kernel support estimation"** (JKSE) [Bócsi et al., 2011b], modelăm funcţia de energie ca neg-probabilitatea comună a datelor $\boldsymbol{x}$ şi $\boldsymbol{\theta}$, *i.e.*,

$$E(\boldsymbol{x}, \boldsymbol{\theta}) \quad \stackrel{\circ}{=} \quad -p(\boldsymbol{x}, \boldsymbol{\theta}). \tag{13}$$

JKSE modelează distribuţia comună a datelor de intrare şi ieşire ca un model log-linear al o funcţiei de feature. După simplificări, predicţia pentru cinematica inversă arată astfel:

$$f^{-1}(\mathbf{x}) \quad = \quad \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta}}{\operatorname{argmax}} \ \boldsymbol{w}^\top \phi(\boldsymbol{x}, \boldsymbol{\theta}),$$

unde $\boldsymbol{w}$ sunt parametri care generează distribuţia $p(\boldsymbol{x}, \boldsymbol{\theta})$ care explică datele $\boldsymbol{D} = \{(\boldsymbol{x}_i, \boldsymbol{\theta}_i)\}_{i=1}^m$ cel mai bine. Valorile parametrilor $\boldsymbol{w}$ sunt obţinute folosind maşini cu suport vectorial cu o singură clasă (one-class support vector machines). Folosind metoda *"structured output Gaussian processes"* (SOGP) [Bócsi et al., 2011a], funcţia de energie $E(\cdot, \cdot)$ este valoarea posterioară medie a unui GP negativă, *i.e.*,

$$E(\boldsymbol{x}, \boldsymbol{\theta}) \quad \stackrel{\circ}{=} \quad -\mu_{(\boldsymbol{x}, \boldsymbol{\theta})}. \tag{14}$$

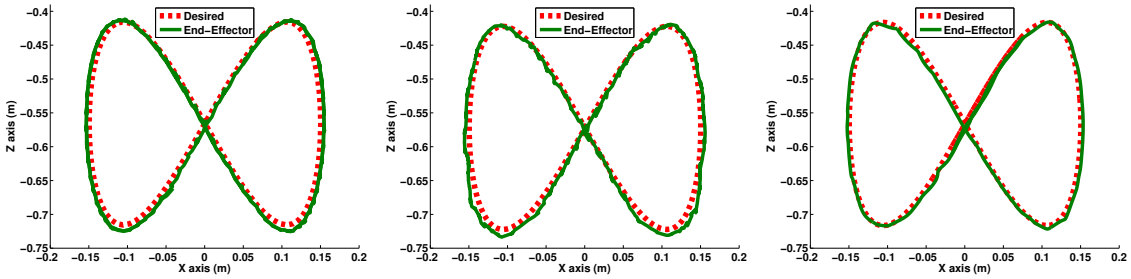
Datele de intrare la GP sunt datele comune de intrare şi ieşire reprezentate de o funcţie $\phi(\mathbf{x}, \theta)$ şi datele de ieşire au valoarea $\mathbf{1}$ la toate datele. O astfel de mulţime de date poate conduce la overfitting; pentru evitarea acestui fenomen prior puternic trebuie aplicat: în cele ce urmează folosim un prior zero pentru a păstra notaţiile simple. Valoarea medie de posterior arată astfel:

$$\mu_{(\boldsymbol{x}, \boldsymbol{\theta})} \quad = \quad \boldsymbol{k}_{(\boldsymbol{x}, \boldsymbol{\theta})}^\top (\boldsymbol{K} + \sigma_0^2 \boldsymbol{I}_{\boldsymbol{m}})^{-1} \mathbf{1},$$

unde $\boldsymbol{K} \in \mathbb{R}^{m \times m}$ cu $\boldsymbol{K}^{ij} = k((\boldsymbol{x}_i, \boldsymbol{\theta}_i), (\boldsymbol{x}_j, \boldsymbol{\theta}_j))$, $\boldsymbol{k}_{(\boldsymbol{x}, \boldsymbol{\theta})} \in \mathbb{R}^{m \times 1}$ cu $\boldsymbol{k}_{(\boldsymbol{x}, \boldsymbol{\theta})}^i = k((\boldsymbol{x}_i, \boldsymbol{\theta}_i), (\boldsymbol{x}, \boldsymbol{\theta}))$, $k_{(\boldsymbol{x}, \boldsymbol{\theta})(\boldsymbol{x}, \boldsymbol{\theta})} = k((\boldsymbol{x}, \boldsymbol{\theta}), (\boldsymbol{x}, \boldsymbol{\theta}))$, $\boldsymbol{I}_{\boldsymbol{m}}$ este matricea identică, σ_0^2 este varianţa zgomotului de măsurare, şi $\mathbf{1}$ este vectorul cu valori 1 de dimensiune m.

A treia abordare, numită "**forward Gaussian process modeling**" (FWGP) [Bócsi et al., 2012], este bazată pe observaţia că modelul cinematic direct – notat cu $f(\cdot)$ – este mult mai uşor de modelat decât funcţia inversă. Bazându-se pe această observaţie, construim funcţia de energie astfel încât ea va depinde explicit pe cinematica directă. Odată ce ştim funcţia de cinematică directă, funcţia de energie este definită ca şi distanţa Euclidiană dintre poziţia dorită de efector şi poziţia anticipată de modelul cinematic direct, *i.e.*,

$$E(\boldsymbol{x}, \boldsymbol{\theta}) \quad \stackrel{\circ}{=} \quad \|\boldsymbol{x} - f(\boldsymbol{\theta})\|^2. \tag{15}$$



(a) JKSE. (b) SOGP. (c) FWGP.

Figura 2: Rezultatele urmăririi formei opt cu învăţare off-line. Se observă o precizie bună cu fiecare model.

Modelăm funcţia de cinematică directă cu un GP. Fiind dat datele de instruire $\boldsymbol{D} = \{(\boldsymbol{x}_i, \boldsymbol{\theta}_i)\}_{i=1}^m$ cu datele de intrare $\boldsymbol{\theta}_i$ şi de ieşire $\boldsymbol{x}_i$, predicţia pentru o nouă $\boldsymbol{\theta}$ are o distribuţie gaussiană cu valoare de medie $\mu_{\boldsymbol{\theta}}$

$$\mu_{\boldsymbol{\theta}} \quad = \quad \sum_{i=1}^{m} \alpha^i k(\boldsymbol{\theta}, \boldsymbol{\theta}_i) = \boldsymbol{k}_{\boldsymbol{\theta}}^\top \boldsymbol{\alpha}, \tag{16}$$

unde $k_{\boldsymbol{\theta}\boldsymbol{\theta}} = k(\boldsymbol{\theta}, \boldsymbol{\theta})$, $\boldsymbol{k}_{\boldsymbol{\theta}} \in \mathbb{R}^{m\times 1}$ este un vector cu elemente $\boldsymbol{k}_{\boldsymbol{\theta}}^i = k(\boldsymbol{\theta}, \boldsymbol{\theta}_i)$ şi $\boldsymbol{K} \in \mathbb{R}^{m\times m}$ este o matrice cu elemente $\boldsymbol{K}^{ij} = k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$. Funcţia $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ este un kernel şi $\boldsymbol{\alpha} \in \mathbb{R}^{m\times 1}$, unde $\boldsymbol{\alpha} = (\boldsymbol{K} + \boldsymbol{I}_m \sigma_0^2)^{-1} \boldsymbol{x}$ sunt parametri GP-ului. Valoare medie posterioară a procesului gaussian este folosită pentru predicţia modelului cinematic direct din equaţia (16), *i.e.*, $f(\boldsymbol{\theta}) = \mu_{\boldsymbol{\theta}}$.

Gradientele pentru funcţiile de energie din ecuaţia (13), ecuaţia (14), şi ecuaţia (15) au forme analitice, deci, căutarea gradient din ecuaţia (12) poate fi făcută eficient.

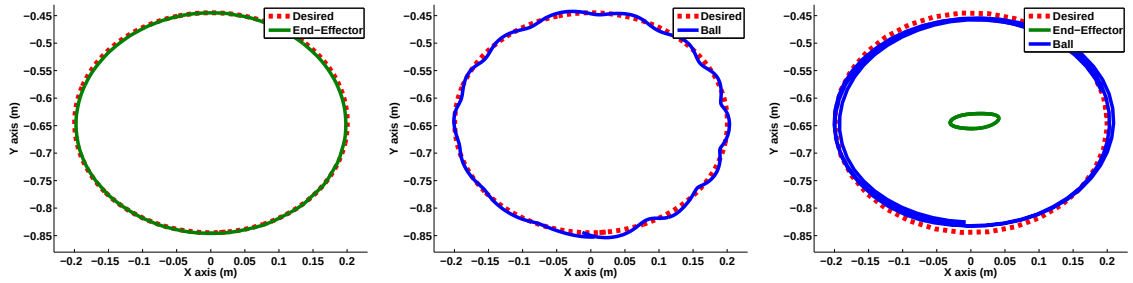
3.3 Experimente

Am evaluat empiric metodele prezentate pentru problema de control de urmărire. Algoritmul a fost aplicat pentru a învăţa cinematica inversă a robotului Barrett WAM [Bócsi et al., 2011b] şi pentru a urmări o figură de opt cu setări diferite. Rezultatele urmăririi sunt prezentate pe figura 2, unde se vede că o precizie bună a fost obţinută. Experimentele arată că FWGP rezultă în modele mai precise decât JKSE sau SOGP care converg la precizia modelului analitic. Rezultatul este puţin surprinzător în lumina numărul punctelor pe care predicţia era bazată. Modelul JKSE a fost bazat pe 8456 puncte, SOGP pe 200 puncte şi FWGP pe 31 puncte.

Într-un experiment diferit, am modificat modelul simulat al robotului făcând-l mai complex prin fixarea unei mingi pe efectorul braţului de robot cu o coardă de 20 cm. Mişcarea oscilatoare a mingii a rezultat într-un sistem neliniar. Cu aceste setări, am făcut control de urmărire când poziţia mingii a fost considerată în loc de poziţia efectorului. Problema a fost de a urmări un circ cu o rază de 20 cm (figura 3a) pe planul orizontal. Experimentul a fost efectuat cu două setări diferite: (1) când punctul de destinaţie a mişcat încet (o rotaţie a fost făcută în 24 de secunde) şi (2) când punctul de destinaţie a mişcat rapid (o rotaţie a fost făcută în 0.62 de secunde).

În primul caz, FWGP a învăţat să-şi mişte efectorul deasupra cercului dorit în timp ce mingea se mişca de-a lungul traiectoriei dorite. Viteza efectorului era aceeaşi ca şi viteza mingii. Pentru a învăţa un model capabil de precizia din Fig 3a şi 3b aveam nevoie de patru minute de interacţiune, iar modelul GP era bazat pe 20-25 de puncte. În al doilea caz efectorul s-a mişcat cu viteză mai mare pe forma circulară, şi mingea a mişcat pe de-a lungul traiectoriei dorite (Fig. 3c). În acest experiment, după 20 de minute de interacţiune, modelul GP a folosit $13-15$ puncte. Comparaţie nu am putu să facem deoarece la această viteză ar fi deteriorat robotul. Precizăm faptul că acelaşi valori ale parametrilor au fost folosiţi în ambele experimente, deci comportamentul adaptiv depinde slab pe hyper-parametri modelului GP. În primul caz, FWGP a considerat mişcarea oscilatorie a mingii ca zgomot, iar în al doilea caz forţa centrifugală a fost încorporată în modelul GP. Experimentele pot fi vizualizate şi la link-urile următoare:

- http://www.youtube.com/watch?v=o-ib-XNJVaM,
- https://www.youtube.com/watch?v=q4D7rQTbikE,
- http://www.youtube.com/watch?v=nRyWLX8GXq0



(a) Traiectoria efectorului cu mişcare încetă. (b) Traiectoria mingii cu mişcare încetă. (c) Traiectoria efectorului şi al mingii cu mişcare rapidă.

Figura 3: Control de urmărire al unui cerc cu un robot Barrett WAM simulat cu o minge fixată pe efector.

3.4 Învăţarea modelelor de roboţi folosind metode de transfer

Motivaţia folosirii metodelor de transfer vine din învăţarea umană [Bócsi et al., 2013]. O diferenţă fundamentală dintre învăţarea umană şi învăţarea automată este aceea că robotul nu are cunoştinţe a-priori despre lume, în timp ce oamenii folosesc propriile experienţe din trecut. În acest context, deşi task-ul nu a fost efectuat de către actorul uman, abilităţile rezultate din experientele trecute va uşura învăţarea problemei respective.

3.4.1 Metode de învăţare prin transfer în robotică

Scopul a fost de a îmbunătăţi procesul de învăţare al modelului de robot când presupunem că informaţii din experimente trecute pot fi folosite în formă de date adiţionale. Definim o sarcină sursă (source task), o problemă care este deja rezolvată, şi o sarcină ţintă (target task), o problemă care este dificil de învăţat. Considerăm problema când sarcina sursă are datele $\mathbf{D}^s = \{(\theta_i^s, \mathbf{x}_i^s)\}_{i=1}^N$ cu N puncte şi dorim să învăţăm o sarcină ţintă cu date de învăţare $\mathbf{D}^t = \{(\theta_i^t, \mathbf{x}_i^t)\}_{i=1}^K$ cu K puncte.

Ca un prim pas, reducem dimensiunea datelor la aceeaşi dimensiune. În experimentele noastre am folosit metoda *"principal component analysis"* (PCA) pentru reducerea dimensiunii. Prin aplicarea metodei PCAse presupune o proiecţie lineară dintre spaţiile cu dimensiuni mici şi mari. Mai întâi, centrăm datele, adică scădem valoare medie, şi le împărţim cu deviaţia, ajungând la:

$$\begin{aligned} \mathbf{s} &= \mathbf{B}_s(\mathbf{d}^s - \mu_s) \\ \mathbf{t} &= \mathbf{B}_t(\mathbf{d}^t - \mu_t), \end{aligned}$$

unde $\mathbf{d}^s \in \mathbf{D}^s$ şi $\mathbf{d}^t \in \mathbf{D}^t$ sunt datele sarcinii sursă şi sarcinii ţintă, iar $\mathbf{s} \in \mathbf{M}^s$ şi $\mathbf{t} \in \mathbf{M}^t$ sunt punctele respective din varietatea cu dimensiune redusă. Valorile $\mu_s = \mathbf{E}\{\mathbf{D}^s\}$ şi $\mu_t = \mathbf{E}\{\mathbf{D}^t\}$ sunt valorile medii ale datelor originale. Matricele $\mathbf{B}_s$ şi $\mathbf{B}_t$ sunt matrice de transformare astfel încât varianţele mulţimilor $\mathbf{M}^s$ şi $\mathbf{M}^t$ să fie maximizaţi. Pentru detalii a se consulta [Bócsi et al., 2012].

În pasul al doilea modelăm funcţia dintre cele două varietăţi ca o proiecţie lineară $f : \mathbf{M}^s \rightarrow \mathbf{M}^t$, cu

$$f(\mathbf{s}) = \mathbf{A}\mathbf{s}, \tag{17}$$

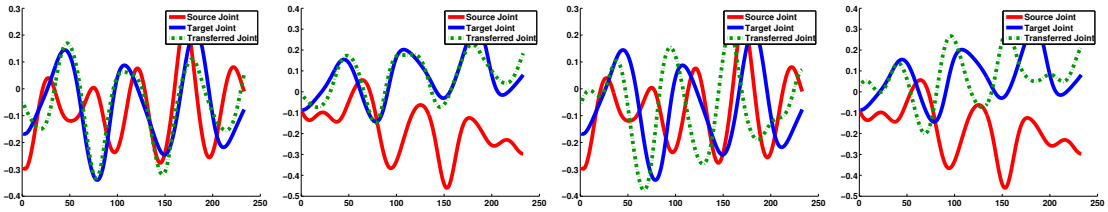
unde $\mathbf{A} \in \mathbb{R}^{J \times J}$ este o matrice de transformare cu dimensiune J. Definim două scenarii de aliniere.

În primul scenariu, ştim o corespondenţă directă dintre punctele mulţimilor. Prin corespondenţă directă înţelegem că $\mathbf{D}^s$ şi $\mathbf{D}^t$ au acelaşi număr de puncte şi punctele sunt împerecheate. Această setare poate fi folosită când aceeaşi sarcină a fost efectuată în spaţiul sursă şi în spaţiul ţintă.

În al doilea scenariu, nu există nici o corespondenţă dintre punctele mulţimilor $\mathbf{D}^s$ şi $\mathbf{D}^t$. Este posibil că mulţimile nu au aceiaşi număr de puncte, *i.e.*, $|\mathbf{D}^s| \neq |\mathbf{D}^t|$.

- În primul caz – numit **aliniere prin corespondenţă directă** –, presupunem o funcţie lineară şi minimizăm eroarea transformaţiei. Dorim să-l găsim valorile parametrilor $\mathbf{A}$ din equaţia (17) astfel încât expectanţa de eroare să fie minimizată, *i.e.*,

$$\mathbf{A} = \underset{\mathbf{A}}{\operatorname{argmin}}\, \mathbf{E}\left\{(\mathbf{t} - \mathbf{A}\mathbf{s})^\top (\mathbf{t} - \mathbf{A}\mathbf{s})\right\},$$



(a) Joint 1. (b) Joint 3. (c) Joint 1. (d) Joint 3.

Figura 4: Rezultate pentru aliniere cu corespondenţă directă şi aliniere dură.

unde $\mathbf{E}\{\cdot\}$ notează operatorul de expectanţă. Soluţia pentru matricea A are următoarea formă:

$$\mathbf{A} \quad = \quad \mathbf{\Sigma}_{ss}^{-1}\mathbf{\Sigma}_{ts}, \tag{18}$$

unde $\mathbf{\Sigma}_{ss}$, $\mathbf{\Sigma}_{tt}$, şi $\mathbf{\Sigma}_{ts}$ sunt matrice de covarianţă.

- În cazul al doilea – numit **aliniere dură** –, distanţa dintre distribuţiile definite de punctele mulţimilor $\mathbf{M}^s$ şi $\mathbf{M}^t$ este minimizată. În ce urmează presupunem că datele au o distribuţie Gaussiană, iar noi minimizăm distanţa dintre două distribuţii Gaussiene $p(\mathbf{M}^s)$ şi $p(\mathbf{M}^t)$. Definită ca şi divergenţa Kullback-Leibler, distanţa dintre două Gaussiene are formă analitică. Ca rezultat, o matrice $\mathbf{A}$ care minimizează această distanţă este soluţia ecuaţiei următoare:

$$\mathbf{\Sigma}_{tt} \quad = \quad \mathbf{A}\mathbf{\Sigma}_{ss}\mathbf{A}^\top.$$

Această expresie este pătratică în $\mathbf{A}$ şi nu are o soluţie unică. O matrice $\mathbf{A}$ care este soluţie a equaţiei precedente poate fi obţinută folosind decompoziţia de valoare proprii a matricelor de covarianţă [Trefethen and Bau, 1997]. Soluţia propusa arată astfel:

$$\mathbf{A} \quad = \quad \mathbf{U}_t\mathbf{\Lambda}_t^{1/2}\mathbf{\Lambda}_s^{-1/2}\mathbf{U}_s^\top,$$

unde $\mathbf{U}_s$ şi $\mathbf{U}_t$ sunt matrice de rotaţie (*i.e.*, $\mathbf{U}_s\mathbf{U}_s^\top = \mathbf{I}$) cu vectori proprii ai $\mathbf{\Sigma}_{ss}$ şi $\mathbf{\Sigma}_{tt}$, iar $\mathbf{\Lambda}_s$ şi $\mathbf{\Lambda}_t$ sunt matrice diagonali cu valori proprii ai $\mathbf{\Sigma}_{ss}$ şi $\mathbf{\Sigma}_{tt}$ respectiv.

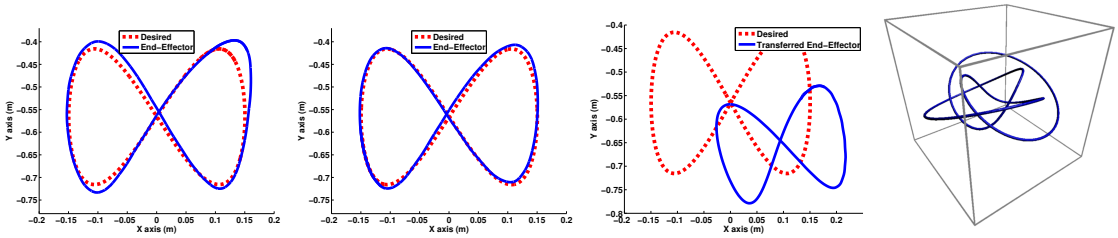
3.4.2 Experimente

Am condus experimente pe roboţi cu diferite arhitecturi pentru a accentua două proprietăţi ale metodei: (1) pierderea de informaţie indusă de reducţia de dimensiune nu este semnificativă şi (2) puterea expresivă a funcţiei lineare este suficientă pentru a obţine aliniere eficientă.

Experimentul a fost efectuat cu un simulator al braţului Sarcos Master cu opt grade de libertate, respectiv cu un simulator al braţului Barrett WAM cu şapte grade de libertate. Scopul în ambele probleme au fost urmărirea unui nod trefoil (figura 5d) cu efectorul roboţilor. Am folosit algoritmul de control analitic al roboţilor pentru a colecta date. După urmărirea figurii cu amândoi roboţi pe timp de un minut, aveam puncte cu corespondenţă directă. Am folosit metoda de aliniere cu corespondenţă directă pe aceste date. Figura 4a şi figura 4b arată că după estimarea matricei $\mathbf{A}$ din equaţia (18), am transferat cu succes (verde) datele de la sarcina sursă (roşu) la sarcina ţintă (albastru). Pentru a vedea câte informaţii pot fi păstrate cu metoda de aliniere dură am aplicat şi metoda respectivă. Rezultatele sunt prezentate pe figura 4c. Se vede că transformarea nu este precisă dar valoarea media şi varianţa sunt păstrate, cum era de aşteptat.

În experimentul următor, am transformat direct o figură de la braţul Master la braţul Barret prin folosirea matricei obţinută din experimentul precedent. Am urmărit o figură de opt care a fost plasată în interiorul spaţiului definit de nodul trefoil cu braţul Master. După transformarea coordinatelor de articulaţii şi urmărirea acestei coordinate cu braţul Barrett, am obţinut figura prezentată pe figura 5c. Prezentarea figurii dorite pe figura 5c este înşelătoare deoarece nu există o figura transformată *corect*. Am definit figura dorită ca şi figura urmărită de controlerul analitic al braţului Barrett cu postura iniţială asemănătoare cu cea folosită pentru nodul trefoil.

În experimentul final, am folosit aceleaşi arhitecturi de roboţi pentru sarcina sursă şi sarcina ţintă ca şi în experimentul anterior. Pentru sarcina sursă am folosit datele colectate în experimentul precedent. Sarcina ţintă era de a accelera învăţarea a funcţiei de cinematică directă a braţului Barrett. Modelul de cinematica directă a



(a) După transfer de date *fără* învăţare adiţională.

(b) După transfer de date *cu* învăţare adiţională.

(c) Transfer de la braţ Master la braţ de Barrett.

(d) Nod trefoil.

Figura 5: Rezultate de control de urmărire după trei secunde de interacţiune şi folosirea metodei propuse.

fost aproximat folosind procese Gaussiene şi acest model a fost folosit pentru control de urmărire. Fără a folosi metode de învăţare prin transfer robotul are nevoie de timp între 20 de secunde şi patru minute pentru a învăţa acest model. După numai trei secunde de mişcări cvasi-stohastice am folosit metoda de aliniere dură. Am oprit procesul de învăţare după trei secunde şi am aplicat metoda noastră. Figura 5a prezintă figura de opt obţinută. Forma figurii de opt nu este perfectă, dar a fost obţinută după numai trei secunde de interacţiune. Am repetat experimentul dar acum procesul de învăţarea nu a fost oprit după trei secunde ci numai datele de la braţul Master au fost folosite. Figura 5b prezintă că dacă procesul de învăţare nu este oprit o figura foarte precisă poate fi obţinută.

Din articolele de mai sus a rezultat teza de doctorat a lui Botond Bócsi, membru în echipa proiectului [Bócsi, 2012]. Teza a avut titlul *"Model Learning for Robot Control"* şi a fost susţinut cu succes în 23 noiembrie 2012.

4 Studiul modelelor cu zgomot de intrare

În general într-o problemă de regresie se consideră zgomot în date de ieşire (label noise). În această cercetare am considerat problema regresiei când este zgomot şi în datele de intrare (input noise).

Fiind dat datele $\mathbf{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, unde $\mathbf{x}_i \in \mathbb{R}^d$ şi $y_i \in \mathbb{R}$, *i.e.*,

$$y = \tilde{y} + \epsilon_y \qquad\qquad \mathbf{x} = \tilde{\mathbf{x}} + \epsilon_x,$$

unde y este ieşirea observată, $\tilde{y}$ este ieşirea reală, $\epsilon_y \sim \mathcal{N}(0, \sigma_y^2)$ este procesul de zgomot, $\mathbf{x}$ data de intrare observată, $\tilde{\mathbf{x}}$ este data reală de intrare, şi $\epsilon_x \sim \mathcal{N}(0, \mathbf{\Sigma})$ procesul de zgomot de intrare.

4.1 Modele cu zgomot de intrare corectate de matricea Hesse

Într-o primă abordare [Bócsi and Csató, 2013] am folosit expansiunea în şirul Taylor a funcţiei de regresie. Folosind proprietatea că procesul de zgomot este aditiv şi Gaussian cu valoare de medie zero am ajuns la următoarea expresie pentru funcţia de regresie corectată:

$$f(\mathbf{x}) \quad = \quad g(\mathbf{x}) - \frac{1}{2}\sigma^\top H_g(\mathbf{x})\sigma, \tag{19}$$

unde σ este varianţa zgomotului de intrare, $g()$ este o funcţie de regresie arbitrară, $H_g()$ este matricea Hesse a funcţiei şi $f()$ este funcţia de regresie corectată.

Am condus experimente pe funcţii unidimensionale şi multidimensionale. Ca funcţie de regresie am folosit procese Gaussiene (GP) şi reţele neuronala (NN) şi versiunea lor corectată de matricea Hesse (+H). Rezultatele sunt afişate pe figura 6. Aici se vede ca îmbunatăţirea impusă de metoda propusă este semnificativă. Mai mult, această îmbunătăţirea devine mult mai vizibilă, când varianţa zgomotului creşte.

Rezultatele experimentelor pe date multidimensionale sunt prezentate în tabelul 1. De aici reiese că îmbunătăţirea impusă de metoda noastră nu este atât de relevantă în acest caz. O explicaţie posibilă este că în teorie preferăm funcţii de regresie lineare pentru a evita over-fittingul. Iar funţiile lineare au derivată de ordin doi zero. Astfel corectare cu matricea Hesse devine nesemnificativ. Această proprietatea nu este a caracteristică a metodei noastre ci un fenomen general a datelor multidimensionale.

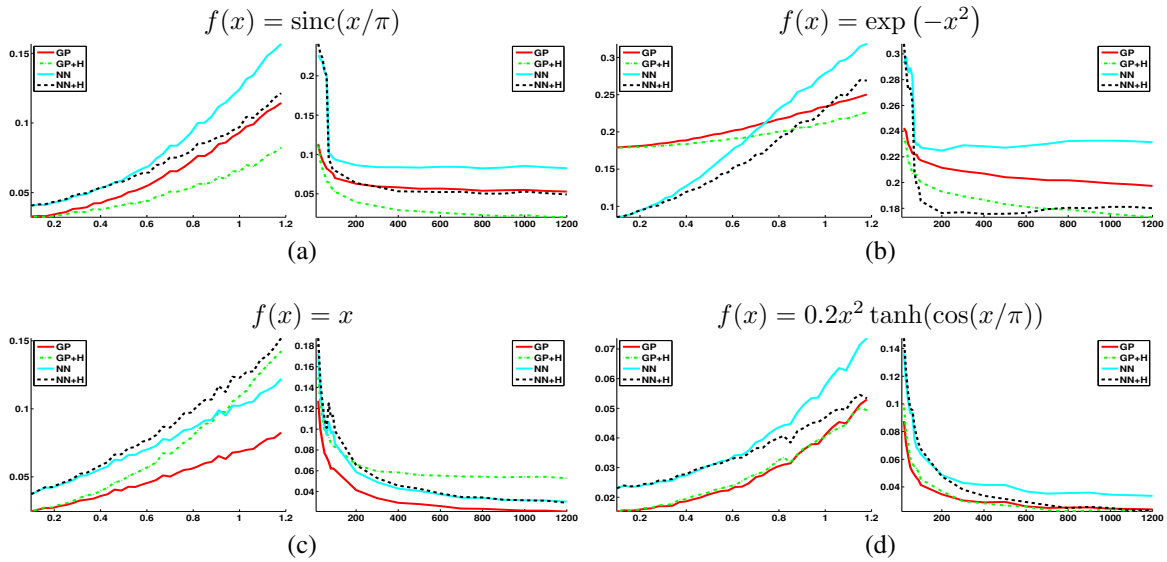


Figura 6: Performanţa metodei SIMEX pe date artificiale.

Nume data-set	GP	GP+H	NN	NN+H	Zgomot (σ)	Dim.	Mărime
Boston housing ($1000s)	2.2271	2.2271	3.4819	3.4819	0.1	13	506
Concrete (MPa)	4.1281	4.1280	6.1865	6.1864	1	8	1030
Barrett WAM 1 (mm)	2.9272	2.9242	2.8726	2.8488	0.01	4	1000
Barrett WAM 2 (mm)	13.707	13.731	12.439	9.3524	0.1	4	1000
CPU performance	17.762	17.763	16.846	16.846	5	7	209
Auto MPG (mpg)	4.0301	4.0322	2.2990	2.2988	3	7	301

Tabela 1: Performanţa metodei SIMEX pe date multidimensionale.

4.2 Simulaţie-extrapolare pentru procese Gaussiene

În a doua abordare [Bócsi et al., 2014b] idea principală este să *generăm* mai multe seturi de date prin adăugarea unui zgomot de intrare cu varianţă controlată:

$$\varphi_\lambda(x) \sim f(x + \sqrt{1+\lambda}\,\epsilon_x), \tag{20}$$

unde $\varphi_\lambda(x)$ este funcţia de regresia bazată de date de intrare când a fost adăugat zgomot cu varianţă λ. Putem observa, că $\varphi_0(x)$ reprezintă funcţia originală când nu a fost adăugat zgomot controlat. Astfel măsurăm efectul zgomotului adăugat şi după câteva simulări de adăugare de zgomot de intrare facem o extrapolare unde nu ar fi zgomot de intrare, $\varphi_{-1}(x)$:

$$\varphi_3, \varphi_2, \varphi_1, \varphi_0 \quad \rightarrow \quad \varphi_{-1}. \tag{21}$$

O ilustraţie a metodei propusă este prezentată pe figura 7.

La experimentele conduse am folosit procese Gaussiene pentru funcţia de regresie. O consecinţe a acestei alegere este că caracteristica probabilistică a proceselor Gaussiene pot fi păstrată, astfel după extrapolare obţinem o distribuţie asupra funcţiilor de regresie. O ilustrare despre îmbunătăţirea impusă de metodă este prezentată pe figura 8.

5 Metode hashing pentru căutare rapidă

Căutările cu metoda kNN – en "k-nearest neighbor" – sunt metode neparametrice foarte performante, fiind demonstrată consistenţa în limita unei baze de date cu număr infinit de elemente. Găsirea vecinilor cei mai apropiaţi este o problemă importantă în multe domenii: găsim aplicaţii importante în regăsirea informaţiei, în prelucrarea

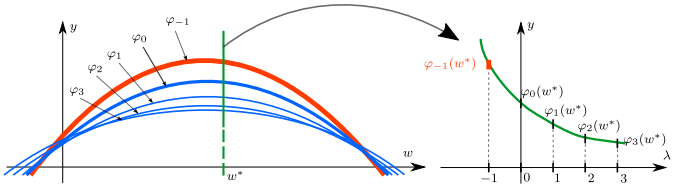


Figura 7: Ilustraţie de simulare-extrapolare. (stânga) φ_λ pentru $\lambda = \{0, 1, 2, 3\}$ cu linii albastre; φ_{-1} este extrapolarea cu roşu. (dreapta) extrapolarea individuală $\varphi_\lambda(w^*)$ – ca a funcţie de λ – pentru data de test w^*.

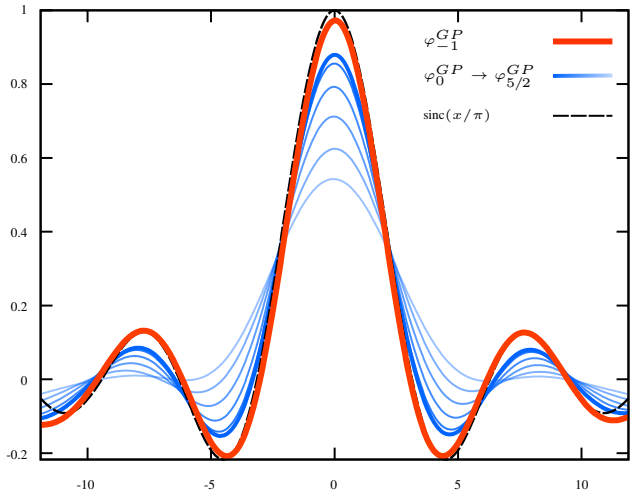


Figura 8: Rezultate pentru funcţia $\mathrm{sinc}(x/\pi)$ cu zgomot de intrare cu varianţă $\Sigma = 0.8$. Curba neagră reprezintă funcţia funcţia realăsinc$(\cdot)$. Curbele albastre $\varphi_0^{GP} \rightarrow \varphi_{5/2}^{GP}$ reprezintă funcţiile de regresie când zgomotul de intrare a fost adăugat. Curba roşie este predicţia de extrapolare φ_{-1}^{GP}.

imaginilor prin calculator (computer vision), în teoria codurilor, în sisteme de recomandare, geometrie computaţională; lista completă este una foarte lungă. Găsirea exactă ai vecinilor cei mai apropiaţi este de complexitate liniară, aceasta deoarece distanţa de la fiecare punct trebuie calculată. Metoda kNN suferă însă de timpul linear, de multe ori fiind nevoie de metode mai rapide – desigur cu o oarecare scădere a performanţei. O clasă de metode având o complexitate subliniară foloseşte coduri binare de hashing: un şir binar *apropiat* este asociat fiecărui punct pentru calcularea rapidă a distanţelor între puncte; iar distanţa Hamming este folosită pentru determinarea rapidă – *dar aproximativă* – a vecinilor.

În Bodó and Csató [2012] am propus o îmbunătăţire a metodei hashing pentru a genera coduri binare, mai exact în metodele numite *"locality-sensitive hashing"* (LSH) bazat pe kerneluri (kLSH), în continuare producem o scurtă descriere a metodei. Algoritmul LSH se bazează pe generarea de vectori aleatori ($\mathbf{r}$), reprezintând hiperplane aleatori,

$$h_{\mathbf{r}}(\mathbf{x}) = \begin{cases} 1, & \text{dacă} \quad \mathbf{r}'\mathbf{x} \geq 0 \\ 0, & \text{în alte situaţii} \end{cases}$$

codul hash de lungime L asociat unui punct $\mathbf{x}$ fiind $[h_{\mathbf{r}_1}(\mathbf{x}), \ldots, h_{\mathbf{r}_L}(\mathbf{x})]'$. Generarea hiperplanurilor aleatori în spaţiul kernel a fost realizată cu o metodă inteligentă, însă problema calculării valorilor de kernel reprezinta problemă. Folosind o metodă simplă de găsire de pre-imagini, am reuşit să îmbunătăţim performaţa algoritmului k-LSH, articolul a fost prezentat la conferinţa IJCNN 2012 [Bodó and Csató, 2012].

Pentru perfecţionarea metodei, în Bodó and Csató [2013] şi Bodó and Csató [2014a] am propus o variantă liniară de spectral hashing – o metodă performantă şi des utilizată pentru căutare ANN (*approximate nearest neighbors*). În spectral hashing calcularea şirurilor binare pentru exemple din mulţimea de training este simplu, însă generalizarea pentru puncte nevăzute devine o problemă dificilă. În articolul nostru propunem a metodă liniară generalizată cu produse scalare în spaţii RKHS pentru calcularea vectorilor de "spectral hashing", unde generarea codurilor binare este la fel de simplu şi pentru punctele de test – adică pentru datele care nu sunt încă incluse în baza de date. Algoritmul *Linear Spectral Hashing* găseşte un set de vectori ortogonali, care sunt vectori normali la hiper-plane de separare cu marjă maximă. Un pseudocod al algoritmului se găseşte în Alg. 1. Lucrarea

Algorithm 1 Linear Spectral Hashing

Input: Datele în $\mathbf{X}$, $\mathbf{D} = \mathrm{diag}(\mathbf{X}'\mathbf{X}\mathbf{1})$, matricele de vectori proprii $\mathbf{V}$ şi $\mathbf{U}$, şi un punct $\mathbf{x}$
Output: Codul $f(\mathbf{x})$

1: Calcularea primelor r vectori proprii a matricei $\mathbf{D}^{-1/2}\mathbf{X}'\mathbf{X}\mathbf{D}^{-1/2}$ sau $\mathbf{X}\mathbf{D}^{-1}\mathbf{X}'$, începând cu a doua cea mai mare valoare proprie:

$$[\mathbf{v}_2,\ \mathbf{v}_3,\ \ldots,\ \mathbf{v}_{r+1}] \quad \text{or}$$
$$[\mathbf{u}_2,\ \mathbf{u}_3,\ \ldots,\ \mathbf{u}_{r+1}]$$

2: Definim funcţia:

$$f_i(\mathbf{x}) = \mathbf{x}'\mathbf{u}_i = \mathbf{x}'\mathbf{X}\mathbf{D}^{-1/2}\mathbf{v}_i e_i^{-1/2}, \quad i = 2, \ldots, r+1$$

unde e_i, $i = 2, \ldots, r+1$ desemnează valorile proprii a matricei $\mathbf{X}\mathbf{D}^{-1}\mathbf{X}'$.

3: Codul lui $\mathbf{x}$ este calculat după cum urmează:

$$f(\mathbf{x}) = [f_2(\mathbf{x}),\ f_3(\mathbf{x}),\ \ldots,\ f_{r+1}(\mathbf{x})]'$$

a fost prezentată la conferinţa ESANN în 2013 [Bodó and Csató, 2013], o versiunea scurtă a apărut în volumul de conferinţei, după aceea am fost invitaţi să publicăm lucrarea în jurnalul Neurocomputing [Bodó and Csató, 2014a].

6 Învăţarea semi-supervizată aplicată la metode de hashing

Învăţarea semi-supervizată este un subdomeniu important al instruirii automate. Din cauza că adnotarea umană a exemplelor de instruire în general solicită expertiză în domeniu, aceasta este costisitoare şi consumatoare de timp. O soluţie este utilizarea numai a unei mici proporţii a datelor etichetate pentru a îmbunătăţi performanţa algoritmului de instruire. Dacă definim învăţarea supervizată ca şi găsirea unei funcţii $\hat{f} : X \to Y$ care aproximează într-un mod optim o funcţie necunoscută $f : X \to Y$, a cărei realizări sunt stocate în exemplele de instruire $(\mathbf{x}, y), \mathbf{x} \in X, y \in Y$, atunci învăţarea semi-supervizată se defineşte prin calculul funcţiei optime folosind un set de date *extins*: $\{(\mathbf{x}_i, y_i) | \mathbf{x}_i \in X, y \in Y, i = 1, 2, \ldots, \ell\} \cup \{\mathbf{x}_j | \mathbf{x}_j \in X, j = \ell+1, \ell+2, \ldots, N\}$. Prima mulţime este mulţimea datelor etichetate, iar a doua este mulţimea datelor neetichetate, $N - \ell = u, \ell \ll u$. În [Bodó and Csató, 2014b] am propus o tehnică de extindere a codurilor hash obţinute printr-o metodă de hashing nesupervizată. Singura prezumpţie este cunoaşterea a câtorva etichete atribuite unor elemente din baza de date. Extinderea este efectuată folosind coduri corectoare de erori şi metode de învăţare semi-supervizată. Algoritmul rezultat este unul generic: atât procedura de generare a codurilor corectoare cât şi metoda de învăţare semi-supervizată pot fi selectate arbitrar. Un cod corector de erori poate detecta $d_{min} - 1$ erori şi poate corecta $\lfloor \frac{d_{min}-1}{2} \rfloor$ erori, unde d_{min} desemnează distanţa minimă Hamming între codurile generate. Un cod corector de erori *favorabil* este caracterizat având rânduri şi coloane bine separate. Pentru a obţine coduri optime în acest sens, putem defini o problemă de optimizare pentru generarea unor astfel de coduri:

$$\min_{\mathbf{C} \in \mathbb{R}^{k \times s}} \quad \sum_{i,j=1}^{k} a_{ij} \|\mathbf{c}_i - \mathbf{c}_j\|^2 \, (= \mathrm{tr}\,(\mathbf{C}'\mathbf{L}\mathbf{C}))$$
$$\text{s.t. } \mathbf{C}'\mathbf{1} = \mathbf{0}, \quad \mathbf{C}'\mathbf{C} = \mathbf{I}$$

unde $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$ desemnează codurile, s este lungimea unui cod, $\mathbf{L} = \mathbf{D} - \mathbf{A}$ este Laplacianul, $\mathbf{A}$ conţine similitudinile între clase, iar $\mathbf{D} = \mathrm{diag}(\mathbf{A}\mathbf{1})$ este matricea de grad a matricei $\mathbf{A}$. Constrângerile sunt introduse pentru a obţine o matrice de coduri echilibrată şi necorelată. Soluţia problemei este dată de vectorii proprii a Laplacianului, şi am testat rezultatele la probleme concrete: în experimente am folosit codurile linear spectral hashing (propuse de tot de noi în [Bodó and Csató, 2013, 2014a]) şi metodele "Laplacian regularized least squares" şi SVM-uri transductive pentru învăţare. Matricele de coduri corectoare de erori au fost generate folosind tehnicile 1-vs-rest, 2-vs-rest şi de optimizare a codurilor cu diferite abordări pentru a calcula similitudinile între clase. Acest rezultat a fost prezentat la conferinţa ESANN în 2014.

În [Bodó and Csató, 2014c] analizăm algoritmul "label propagation" (LP) pentru învăţare semi-supervizată. Algoritmul este unul transductiv şi bazat pe graful de similitudini între exemple. Schema algoritmului este prezen-

Algorithm 2 Label propagation

1: **repeat**
2: $\mathbf{Y} = \mathbf{TY}$
3: (Normalizarea rândurilor din $\mathbf{Y}$.)
4: Restaurarea datelor etichetate.
5: **until** convergenţă

tată în Alg. 2: rândul 2 realizează propagarea etichetelor, rândul 3 este un pas opţional, iar ultimul pas constă ăn *restaurarea* etichetelor ale datelor etichetate, modificate în pasul de propagare. Articolul compară două versiuni al algoritmului LP: diferenţa între cele două variante este în alegerea matricei de tranziţie $\mathbf{T}$. Dacă definim matricea de similitudine între exemple ca $\mathbf{W}$, iar $\mathbf{D} = \mathrm{diag}(\mathbf{W1})$ desemnează matricea de grad, matricea probabilităţilor de tranziţie este $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$. În prima versiune a algoritmului matricea T este $\mathbf{P}'$ (transpusa matricei $\mathbf{P}$), iar în a doua versiune este egală cu $\mathbf{P}$. Deşi diferenţa pare subtilă, outputul algoritmului – cel puţin în ceea ce priveşte formulele – diferă:

$$\begin{aligned} \mathbf{Y}_U &= \mathbf{A}\mathbf{D}_L^{-1}\mathbf{Y}_L \\ \mathbf{Y}_U &= \mathbf{A}\mathbf{Y}_L \end{aligned}$$

Răspunsurile date de cele două algoritmi sunt analizate şi comparate între ele şi în experimente cu date reale. Articolul, de asemenea, încearcă să facă o recomandare, folosirea cărei variantă este mai favorabilă. Articolul a fost trimis spre publicare la Studia Universitatis Babeş–Bolyai, Series Informatica.

7 Concluzii

În urma analizelor făcute apreciem că metodele neparametrice pot fi folosite cu succes în procesarea bazelor de date. Deşi cuantumul de procesare adăugată este mare, dacă modelul este destul de bine specificat, atunci cu alocarea mai multor resurse se pot obţine rezultate superioare comparativ cu algoritmii clasici.

La algoritmi de învăţare prin întăriri am constatat că este important să avem posibilitatea ca datele să fie procesate într-un mod on-line: odată ce datele sosesc, ele pot fi procesate fără a aştepta terminarea procesului de colecţie a acestora. În acest sens este important formalismul din Secţiunea 2.5.2, formula (11), care produce un update rapid şi eficient. În viitorul apropiat vrem să testăm rezultatul la modelele mai complicate din robotică şi să-l publicăm în jurnale de referinţă.

Un al doilea rezultat important este acela al studiului modelelor cu zgomot de intrare: modelul considerat este unul greoi considerând timpul de execuţie. Totodată nu studiul consistenţei metodei SimExGP ar aduce beneficii vizavi de aplicabilitatea metodei SimExGP la sisteme din lumea reală. Considerăm că – de exemplu prin utilizarea calculului variaţional – o să putem să realizăm alte metode de eliminare sau de tratare a zgomotului la punctele de intrare, iar comparaţiile cu metoda SimExGP pot fi utile (modelele derivate bazate pe calculul variaţional de obicei sunt mai rapide; credem că metoda SimExGP este una bună dar lentă, iar accelerarea metodei va fi benefică.

Mulţumiri: prin acesta dorim să mulţumim pentru sprijinul acordat de către UEFISCDI pentru derularea proiectului *PN-II-RU-TE-2011-3-0278*. Ca urmare a finanţării, au fost susţinute două teze de doctorat, dintre care unul s-a realizat în co-tutelă.

Bibliografie

B. Bócsi and L. Csató. Hessian corrected input noise models. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, volume 8131 of *LNCS*, pages 1–8, Sofia, Bulgaria, 2013. Springer. ISBN 978-3-642-40727-7. doi: 10.1007/978-3-642-40728-4_1.

B. Bócsi, L. Csató, and J. Peters. Structured output gaussian processes. Technical report, Babes-Bolyai University, 2011a. URL http://www.cs.ubbcluj.ro/~bboti/pubs/sogp_2011.pdf.

B. Bócsi, D. Nguyen-Tuong, L. Csató, B. Schoelkopf, and J. Peters. Learning inverse kinematics with structured prediction. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 698–703, San Francisco, USA, September 2011b. ISBN 978-1-61284-454-1. doi: 10.1109/IROS.2011.6094666.

B. Bócsi, P. Hennig, L. Csató, and J. Peters. Learning tracking control with forward models. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 259–264, St. Paul, MN, USA, 2012. ISBN 978-1-4673-1403-9. doi: 10.1109/ICRA.2012.6224831.

B. Bócsi, L. Csató, and J. Peters. Alignment-based transfer learning for robot models. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, Dallas, USA, 2013. doi: 10.1109/IJCNN.2013.6706721.

B. Bócsi, L. Csató, and J. Peters. Indirect robot model learning for tracking control. *Advanced Robotics*, 28(9): 589–599, 2014a. doi: 10.1080/01691864.2014.888371.

B. Bócsi, H. Jakab, and L. Csató. Simulation-extrapolation gaussian processes for input noise modeling. In *Proceedings of the 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, Timisoara, Romania, September 2014b.

B. Bócsi. *Model Learning for Robot Control*. PhD thesis, Babeş-Bolyai University of Cluj-Napoca, Faculty of Mathematics and Computer Science, 2012.

B. A. Bócsi, H. S. Jakab, and L. Csató. Simulation extrapolation gaussian processes for input noise modeling. In *16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2014, Timisoara, Romania, September 21-25, 2014*, pages to–be–published, 2014.

Z. Bodó and L. Csató. Improving kernel locality-sensitive hashing using pre-images and bounds. In *Proceedings of the 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 2710–2717, 2012.

Z. Bodó and L. Csató. Linear spectral hashing. In *Proceedings of the 21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pages 303–308, 2013.

Z. Bodó and L. Csató. Linear spectral hashing. *Neurocomputing*, 141:117–123, 2014a.

Z. Bodó and L. Csató. Augmented hashing for semi-supervised scenarios. In *Proceedings of the 22nd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pages 53–58, 2014b.

Z. Bodó and L. Csató. A note on label propagation for semi-supervised learning. *Studia Universitatis Babeş–Bolyai, Series Informatica*, 2014c. (submitted).

S. J. Bradtke, A. G. Barto, and P. Kaelbling. Linear least-squares algorithms for temporal difference learning. In *Machine Learning*, pages 22–33, 1996.

L. Csató and M. Opper. Sparse on-line Gaussian Processes. *Neural Computation*, 14(3):641–669, 2002.

M. P. Deisenroth, C. E. Rasmussen, and J. Peters. Gaussian process dynamic programming. *Neurocomputing*, 72 (7-9):1508–1524, 2009. ISSN 0925-2312. doi: http://dx.doi.org/10.1016/j.neucom.2008.12.019.

M. Ghavamzadeh and Y. Engel. Bayesian policy gradient algorithms. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *NIPS '07: Advances in Neural Information Processing Systems 19*, pages 457–464, Cambridge, MA, 2007. MIT Press.

H. Jakab and L. Csató. Improving Gaussian process value function approximation in policy gradient algorithms. In T. Honkela, W. Duch, M. Girolami, and S. Kaski, editors, *Artificial Neural Networks and Machine Learning – ICANN 2011*, volume 6792 of *Lecture Notes in Computer Science*, pages 221–228. Springer, 2011. ISBN 978-3-642-21737-1.

H. Jakab and L. Csató. Reinforcement learning with guided policy search using Gaussian processes. In *The 2012 International Joint Conference on Neural Networks (IJCNN),Brisbane, Australia, June 10-15, 2012*. IEEE, 2012. ISBN 978-1-4673-1488-6.

H. Jakab and L. Csató. Manifold-based non-parametric learning of action-value functions. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN)*, pages 579–585, Bruges, Belgium., 2012. UCL, KULeuven, (c)Ciaco. ISBN 978-2-87419-047-6.

H. S. Jakab. *Intelligent Models for Robotic Behavior, Decision Making and Environment Interaction*. PhD thesis, Joint PhD degree between: the Faculty of Computer Science, Eötvös Loránd University of Budapest, Hungary and Faculty of Mathematics and Computer Science, Babeş-Bolyai University of Cluj-Napoca, Romania, 2012.

H. S. Jakab and L. Csató. Novel feature selection and kernel-based value approximation method for reinforcement learning. In *Artificial Neural Networks and Machine Learning – ICANN*, volume 8131 of *Lecture Notes in Computer Science*, pages 170–177. Springer, 2013. ISBN 978-3-642-407277.

H. S. Jakab and L. Csató. Sparse approximations to value functions in reinforcement learning. In P. Koprinkova-Hristova, editor, *Springer Series in Bio-/Neuroinformatics, Artificial Neural Networks*, volume 4, pages 295–315. Springer International Publishing Switzerland, 2014. doi: 10.1007/978-3-319-09903-3_14.

M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, 1994.

C. E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

L. N. Trefethen and D. Bau, III. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997. ISBN 0-89871-361-7.