Optimizare, învăţare automată, aplicaţii

Radu-Emil Precup
Universitatea Politehnica Timişoara
Departamentul de Automatică şi Informatică Aplicată
http://www.aut.upt.ro/~rprecup/

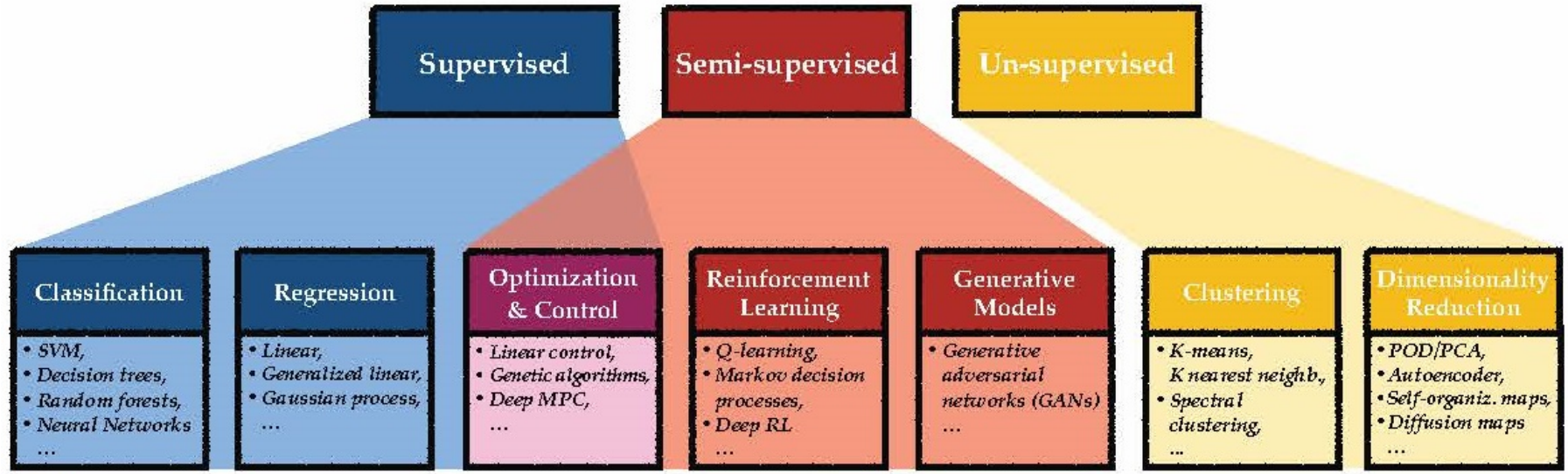
Bibliografie

R.-E. Precup, Tehnici de optimizare, curs, Universitatea Politehnica Timişoara, 2021.

R.-E. Precup, E.-L. Hedrea, R.-C. Roman, E. M. Petriu, A.-I. Szedlak-Stinean, C.-A. Bojan-Dragos, Experiment-based approach to teach optimization techniques, IEEE Transactions on Education, vol. 64, no. 2, pp. 88-94, May 2021.

S. L. Brunton, B. R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, Annual Review of Fluid Mechanics, vol. 52, pp. 477-508, Jan. 2020.

Start: fig. 1 din (Brunton et al., 2020) – clasificarea algoritmilor de învăţare automată în funcţie de informaţiile disponibile (cantitate, tip) în procesul de învăţare:



Definirea unei probleme de optimizare

În sens larg, ***optimizare*** = acţiunea de stabilire, pe baza unui criteriu prestabilit, a celei mai bune decizii (soluţii) într-o situaţie dată când sunt posibile mai multe decizii, precum şi acţiunea de implementare a deciziei stabilite împreună cu rezultatul ei.

În sens restrâns, ***optimizare*** = doar acţiunea de stabilire a celei mai bune decizii (soluţii), numită **decizie optimală** (**soluţie optimală**).

Definiţia implică existenţa a **trei componente:**

(i) o problemă tehnică referitoare la calculul matematic al unei soluţii,

(ii) mai multe soluţii pentru aceeaşi problemă,

(iii)un criteriu de selectare a soluţiei optime (criteriul de optimizare).

Enunţul unei probleme de optimizare (PO) în sens restrâns trebuie să conţină două *elemente*:

A) modelul mediului la care se referă situaţia dată;

B) criteriul de optimizare.

Rezolvarea unei PO presupune existenţa unui al treilea element:

C) metoda de optimizare.

A) Modelul mediului – redă procesul cauzal din cadrul mediului la care se referă PO. Este elementul pe baza căruia sunt estimate efectele diferitelor decizii care pot fi luate în considerare.

Conţine patru categorii de relaţii şi este un sistem de relaţii care caracterizează interdependenţele între variabile. **Variabilele** care intervin în expresiile criteriului = **variabile decizionale** sau **elemente programabile** ale problemei.

Prezentarea celor patru categorii de *relaţii* – în legătură cu optimizarea sistemelor (proceselor):

1) Ecuaţiile procesului, în diverse forme cunoscute din domeniul sistemelor dinamice, în care apar cele trei tipuri de *mărimi*:

a) mărimi *variabile în timp*: de intrare (comandă), $\underline{u} \in U \subseteq R^m$, de stare, $\underline{x} \in X \subseteq R^{ns}$, de ieşire, $\underline{y} \in Y \subseteq R^p$;

b) variabila independentă *timp*:

$t \in T_{0f} \subseteq R$ - pentru sisteme cu timp continuu (SC),

$t \in T_{0f} = \{t_{k0}, t_{k0+1}, \ldots, t_{kf}\} = \{t_k \mid k = k_o, \ldots, k_f \subset Z\} \subset R$ - pentru sisteme cu timp discret (SD).

Domeniul T_{0f} = **interval de optimizare** sau **orizont de timp**. În anumite probleme orizontul de timp este *finit*: $T_{of} = [t_0, t_f] \subset R$, cu t_0 şi t_f – finite pentru SC, respectiv k_0 şi k_f – finite pentru SD, iar în altele este *infinit*: $T_{of} = [t_0, \infty) \subset R$, cu t_0 – finit pentru SC, respectiv k_0 – finit, k_f – infinit pentru SD; t_0 (t_{k0}) se numeşte **moment iniţial** şi t_f (t_{kf}) se numeşte **moment final**.

c) mărimi *constante în timp*: parametrii constructivi sau de proiectare, $\underline{p}_c \in P_c \subseteq R^{qc}$, parametrii de acordare sau funcţionali, $\underline{p}_a \in P_a \subseteq R^{qa}$.

2) Domeniile admise pentru mărimile care apar în ecuaţiile procesului – au fost prezentate anterior.

3) Condiţiile iniţiale şi finale care sunt asociate capetelor (bornelor) t_0 (t_{k0}) şi t_f (t_{kf}) ale orizontului de timp pe care a fost definită PO – se referă de obicei la mărimile de stare:

$\underline{x}(t_0) = \underline{x}_0, \ \underline{x}(t_f) = \underline{x}_f$

şi, mai rar, la mărimile de ieşire:

$\underline{y}(t_0) = \underline{y}_0, \ \underline{y}(t_f) = \underline{y}_f,$

domeniile corespunzătoare având expresiile:

$\underline{x}_0 \in X_0 \subseteq X, \underline{x}_f \in X_f \subseteq X, \underline{y}_0 \in Y_0 \subseteq Y, \underline{y}_f \in Y_f \subseteq Y$

şi purtând denumirile: varietate (domeniu) **iniţial** (**de lansare**) şi varietate (domeniu) **final** (**ţintă**).

4) Condiţiile suplimentare impuse mărimilor care apar în ecuaţiile procesului – datorate particularităţilor situaţiei în care se găseşte mediul considerat.

Sunt exprimate printr-un sistem de ecuaţii algebrice şi / sau diferenţiale (cu diferenţe) şi / sau inecuaţii algebrice şi / sau integrale şi / sau diferenţiale (cu diferenţe), valabil pe întregul orizont de timp T_{0f} sau numai la anumite momente ale acestuia.

Ecuaţiile algebrice menţionate – **restricţii de tip egalitate** (RTE) – pot fi exprimate prin:

$$h_i(v_1, v_2, ..., v_n) = 0,\ i = 1...n_{RTE} \qquad (1.3)$$

în cazul în care este presupus că sunt impuse n_{RTE} RTE. În unele lucrări RTE sunt numite şi **restricţii active**.

Inecuaţiile algebrice menţionate – **restricţii de tip inegalitate** (RTI) – pot fi exprimate prin:

$$l_j \le g_j(v_1, v_2, ..., v_n) \le L_j, \ \ j = 1...n_{RTI} \tag{1.5}$$

în cazul în care este presupus că sunt impuse n_{RTI} RTI, l_j – limitele inferioare admise, L_j – limitele superioare admise.

Dacă este impusă câte o singură limită care uneori poate fi nulă → RTI de forma:

$$g_j(v_1, v_2, ..., v_n) \ge 0, \ \ j = 1...n_{RTI} \tag{1.6}$$

sau de forma:

$$g_j(v_1, v_2, ..., v_n) \le 0, \ \ j = 1...n_{RTI} \ . \tag{1.7}$$

În unele lucrări RTI sunt numite şi **restricţii inactive**, iar expresiile acestora (membrii stângi din (1.6) şi (1.7)) sunt numite **ecuaţii limită**.

În unele lucrări restricţiile sunt numite **constrângeri** (**constraints**).

Numărul n_{RTE} al RTE trebuie să fie strict mai mic decât numărul *n* al variabilelor $\rightarrow$ este necesară impunerea condiţiei:

$$n_{RTE} < n \,. \tag{1.10}$$

Dacă are loc **egalitatea** $n_{RTE} = n$ $\rightarrow$ nu mai are rost rezolvarea PO deoarece pentru respectarea RTE sistemul poate funcţiona numai cu anumite valori fixe ale variabilelor obţinute ca soluţii ale **sistemului algebric (1.3) de $n_{RTE} = n$ ecuaţii cu $n_{RTE} = n$ necunoscute**.

Dacă are loc **inegalitatea** $n_{RTE} > n$ $\rightarrow$ PO devine *în general incompatibilă* pentru că este redusă la rezolvarea sistemului algebric (1.3) la care numărul ecuaţiilor este strict mai mare decât numărul necunoscutelor.

În cazul RTI nu intervin condiţii de tip (1.10) deoarece RTI delimitează anumite domenii în $R^n \rightarrow n_{RTI}$ poate depăşi n.

Mulţime fezabilă sau **mulţime admisibilă** – mulţimea tuturor soluţiilor sistemelor de ecuaţii ale procesului menţionate la 1), care aparţine integral domeniilor admise şi care satisface condiţiile iniţiale, finale şi suplimentare impuse – notată cu Π_{0f} :

$$\Pi_{0f} = \{\underline{v} \in R^n \mid h_i(\underline{v}) = 0,\ i = 1...n_{RTE},\ g_j(\underline{v}) \geq 0,\ j = 1...n_{RTI}\} \qquad (1.11)$$

sau:

$$\Pi_{0f} = \{\underline{v} \in R^n \mid h_i(\underline{v}) = 0,\ i = 1...n_{RTE},\ g_j(\underline{v}) \leq 0,\ j = 1...n_{RTI}\}. \qquad (1.12)$$

În cazul particular al optimizării sistemelor (proceselor), un element al mulţimii fezabile (mulţimii admisibile) – **proces admisibil** sau **proces fezabil** sau **traiectorie globală admisibilă** sau **traiectorie globală fezabilă**.

Un proces admisibil (proces fezabil) – orice soluţie a sistemelor de ecuaţii ale procesului menţionate la 1), care aparţine integral domeniilor admise şi care satisface condiţiile iniţiale, finale şi suplimentare impuse.

Π_{0f} este de data aceasta **mulţimea proceselor admisibile**:

$$\Pi_{0f} = \left\{ \{ \underline{x}(\bullet), \underline{u}(\bullet), ..., \underline{p}_a \} \,|\, \underline{x} \in X, \underline{u} \in U, ..., \underline{p}_a \in P_a \ \forall t \in T_{0f}; \right.$$
$$\underline{x}_0 \in X_0, ..., \underline{y}_f \in Y_f; \ \underline{g}(\bullet,\bullet) = \underline{0}, \ \underline{h}(\bullet,\bullet) \geq \underline{0} \ \forall t \in T_{0f}; \qquad (1.13)$$
$$\left. \underline{g}_0(\bullet_0,\bullet_0) = \underline{0}, \ \underline{g}_f(\bullet_f,\bullet_f) = \underline{0} \right\}.$$

Modelarea mediului la care se referă situaţia dată este deosebit de importantă → efortul principal şi volumul cel mai important de muncă din cadrul unei PO este îndreptat spre cunoaşterea sistemului şi descrierea sa cantitativă.

În practică este recomandat să fie realizat un **compromis rezonabil între complexitatea modelului, precizia acestuia şi uşurinţa de soluţionare a acestuia** deoarece în cadrul celor mai multe metode de optimizare este necesară evaluarea de zeci de ori a modelului matematic asociat funcţiei obiectiv.

B) Criteriul de optimizare – exprimat în general prin **funcţia obiectiv** (**funcţie criteriu**, **funcţie cost**, **funcţie de optimizare**, **funcţie scop**, FO) şi reflectă atitudinea faţă de FO, imprimată de PO.

FO are rol de indicator de performanţă şi serveşte la evaluarea numerică a diferitelor decizii, iar atitudinea trebuie să specifice sensul de variaţie dorit al FO: minimizare sau maximizare.

FO este de regulă o funcţională:

$$J : \Pi_{0f} \to R, \tag{1.14}$$

care asociază fiecărui element al mulţimii fezabile (fiecărui proces admisibil) un număr real prin care este apreciată calitatea absolută a acestui element (proces) în raport cu mulţimea Π_{0f} .

Principii care trebuie respectate la determinarea expresiei matematice a FO, (1), (2), (3):

(1) Principiul unicităţii – formularea PO pe baza utilizării unei singure FO.

Dacă este impusă utilizarea mai multor FO care trebuie optimizate simultan $\rightarrow$ PO asociate acestora se numesc **probleme de optimizare multicriterială** (**optimizare multiobiectiv**, **optimizare vectorială**).

De cele mai multe ori criteriile de optim sunt contradictorii $\rightarrow$ îngreunează alegerea şi aplicarea metodei de optimizare. Abordarea cea mai simplă – exprimarea acestor FO, dacă este posibil, prin una singură, prin intermediul **combinaţiilor liniare ponderate** ale FO componente $\rightarrow$ PO este redusă la una cu un singur obiectiv, în care fiecare criteriu participă în această FO prin înmulţire cu o pondere prestabilită.

(2) Principiul controlabilităţii – utilizarea în cadrul FO a acelor variabile care să poată fi manipulate. Prin intermediul variabilelor decizionale utilizate trebuie să fie asigurată modificarea valorii FO.

(3) Principiul unimodalităţii – existenţa în mulţimea admisibilă a unui singur extrem, care poate fi situat chiar pe frontiera mulţimii admisibile Π_{0f}.

C) Metoda de optimizare – ansamblul de mijloace utilizate pentru determinarea deciziei optimale pe baza modelului mediului şi a FO.

Enunţarea unei PO – formă generală:

$$A: \hat{\underline{v}} = \arg\min_{\underline{v}} J(\underline{v}) = B \text{ s.l.}: \underline{C}, \underline{D}, \tag{1.19}$$

$\underline{v}$ – variabilele problemei (variabilele decizionale, elementele programabile), $\hat{\underline{v}}$ – soluţia problemei (optimul), A se înlocuieşte cu denumirea PO, B se înlocuieşte cu expresia FO, $\underline{C}$ se înlocuieşte cu RTE, $\underline{D}$ se înlocuieşte cu RTI.

„s.l.” – „supusă la”, echivalentul în engleză: „s.t.” (subject to).

Orice problemă de **maximizare poate fi transformată într-o problemă de minimizare** prin schimbarea semnului FO datorită:

$$\max_{\underline{v}} J(\underline{v}) = -\min_{\underline{v}} -J(\underline{v}). \tag{1.20}$$

Într-o PO valoarea FO e mai puţin importantă. Contează doar ca **valoarea FO să fie minimă**.

În unele PO (de exemplu, în probleme de conducere optimală a anumitor procese) FO nu depinde puternic, în zona învecinată optimului, de variabilele problemei (FO are **sensibilitate redusă în raport cu modificările variabilelor**) $\rightarrow$ **în exprimarea cantitativă a termenilor FO nu este necesară o precizie ridicată**, comparabilă celei cerute de ecuaţiile modelului matematic.

Clasificarea PO după **proprietăţile matematice ale funcţiilor**.

Proprietăţi pentru funcţia obiectiv	Proprietăţi pentru funcţiile associate restricţiilor
Funcţie de o variabilă	Fără restricţii
Funcţie liniară	Margini simple pentru variabile
Sumă de pătrate de funcţii liniare	Funcţii liniare
Funcţie pătratică	Funcţii neliniare diferenţiabile
Sumă de pătrate de funcţii neliniare	Funcţii neliniare nediferenţiabile
Funcţie neliniară diferenţiabilă	
Funcţie neliniară nediferenţiabilă	

Probleme de **programare liniară** – atât funcţia obiectiv J cât şi toate restricţiile h_i şi g_j sunt liniare.

Probleme de **programare neliniară** – funcţia obiectiv J sau unele dintre restricţiile h_i sau g_j sunt neliniare.

Programarea liniară şi programarea neliniară sunt grupate în cadrul **programării matematice**.

Aspecte particulare ale problemelor de programare neliniară: **programare pătratică** – FO este pătratică şi toate restricţiile sunt liniare; **programare convexă** – FO este convexă şi toate restricţiile sunt convexe.

În cazul problemelor de optimizare a sistemelor (proceselor), în funcţie de precizarea sau nu a orizontului de timp:

- **probleme de optimizare dinamică** (POD) – precizarea orizontului de timp este fundamentală,

- **probleme de optimizare staţionară** (POS) – precizarea orizontului de timp nu e necesară nici pentru modelul mediului şi nici pentru FO şi în care interesează doar regimul staţionar al sistemului dinamic. Pentru POS, FO sunt de forma:

$$J = \theta(\underline{x}, \underline{u}). \tag{1.57}$$

Problematica rezolvării numerice a problemelor de optimizare fără restricţii

Forma generală a problemelor de optimizare fără restricţii:

$$\text{PFR}: \hat{\mathbf{v}} = \arg \min_{\mathbf{v} \in R^n} J = \theta(\mathbf{v}). \tag{4.1.1}$$

Pentru rezolvarea PFR (4.1.1) prezintă interes numai ***metodele*** numerice ***de descreştere*** (***de relaxare***) – duc la descreşterea valorii funcţiei obiectiv (FO) θ la fiecare iteraţie:

$$\theta(\mathbf{v}^{k+1}) < \theta(\mathbf{v}^k), \ k = 0,1,..., \tag{4.1.2}$$

k – numărul iteraţiei curente.

Structura recurentă a metodelor de relaxare:

$$\mathbf{v}^{k+1} = \mathbf{v}^k + s^k \mathbf{d}^k,\ k = 0,1,..., \tag{4.1.3}$$

vectorul $\mathbf{d}^k \in R^n$ – **direcţia de deplasare** (**de căutare**) din punctul curent $\mathbf{v}^k$, scalarul $s^k > 0$ – **lungimea pasului de căutare** (**de deplasare**).

(4.1.3) – algoritm de învăţare $\rightarrow$ legătură cu învăţarea.

Rescrierea relaţiei (4.1.3):

$$\mathbf{v}^{k+1} - \mathbf{v}^k = \Delta\mathbf{v}^k = s^k \mathbf{d}^k,\ k = 0,1,..., \tag{4.1.4}$$

arată că aproximaţia următoare, $\mathbf{v}^{k+1}$, este obţinută efectuând asupra lui $\mathbf{v}^k$ o corecţie $\Delta\mathbf{v}^k$ determinată exclusiv de direcţia $\mathbf{d}^k$ şi de pasul s^k, adoptate la iteraţia k.

Algoritmii de rezolvare numerică a PFR – etapele 0 – 4:
Fie $\mathbf{v}^0 \in R^n$ o estimaţie iniţială a minimului.

Etapa 0 (iniţializare). Este efectuată iniţializarea $k = 0$.

Etapa 1 (test de convergenţă). Dacă sunt satisfăcute condiţiile de convergenţă $\rightarrow$ algoritmul este terminat cu soluţia $\mathbf{v}^k$.

Altfel, algoritmul continuă cu etapa 2.

Etapa 2 (determinarea direcţiei de căutare). Este calculată direcţia de căutare, $\mathbf{d}^k$.

Etapa 3 (calculul lungimii pasului). Este calculată lungimea pasului $s^k>0$ astfel încât (a se vedea (4.1.2) şi (4.1.3)):

$$\theta(\mathbf{v}^k + s^k \mathbf{d}^k) < \theta(\mathbf{v}^k), \ k = 0,1,\ldots . \qquad (4.1.5)$$

Etapa 4 (actualizarea estimaţiei minimului). Sunt efectuate actualizările: $\mathbf{v}^k \leftarrow \mathbf{v}^k + s^k \mathbf{d}^k$, $k = k+1$ şi urmează salt la etapa 1.

Pentru ***convergenţa*** algoritmului către un minim local $\mathbf{v}^*$ al FO θ trebuie impuse ***condiţii*** atât asupra lui θ cât şi asupra alegerii lui $\mathbf{d}^k$ şi s^k. Condiţiile impuse FO θ:

a) θ este de clasă C^2 (de două ori continuu diferenţiabilă),

b) mulţimea $\{ \mathbf{v}^0 \in R^n \mid \theta(\mathbf{v}) < \theta(\mathbf{v}^0) \}$ este închisă şi mărginită.

Rezolvarea numerică a problemelor celor mai mici pătrate neliniare

În multe aplicaţii practice, FO $\theta(\mathbf{v})$ este o sumă de pătrate de funcţii:

$$\theta(\mathbf{v}) = \frac{1}{2} \| \mathbf{f}(\mathbf{v}) \|^2 = \frac{1}{2} \sum_{i=1}^{m} f_i^2(\mathbf{v}), \tag{4.4.1}$$

în care $\mathbf{f} : R^n \to R^m$, $\mathbf{f}(\mathbf{v}) = [f_1(\mathbf{v}) \dots f_i(\mathbf{v}) \dots f_m(\mathbf{v})]^T$, $\mathbf{v} = [v_1 \dots v_j \dots v_n]^T$.

Valoarea $\| \mathbf{f}(\mathbf{v}) \|$ este numită de regulă *reziduul* în $\mathbf{v}$. Problema minimizării FO din (4.4.1) este numită **problema celor mai mici pătrate neliniare** (PCMMPN):

$$\text{PCMMPN} : \hat{\mathbf{v}} = \arg \min_{\mathbf{v} \in R^n} \theta(\mathbf{v}) = \frac{1}{2} \| \mathbf{f}(\mathbf{v}) \|^2 = \frac{1}{2} \sum_{i=1}^{m} f_i^2(\mathbf{v}). \tag{4.4.2}$$

PCMMPN este formulată în (4.4.2) ca o PFR dar pot fi impuse RTE şi/sau RTI.

PO de tip (4.4.2) sunt utilizate în **estimarea neliniară a parametrilor** ca un caz particular al unor probleme de modelare a sistemelor sau de aproximare a funcţiilor (**model fitting**, **curve fitting**, **data fitting**). În astfel de probleme, dacă $\varphi(\mathbf{v},l)$ este funcţia utilizată drept model matematic, în care l este un parametru independent, atunci funcţiile individuale $f_i(\mathbf{v})$ din (4.4.1) şi (4.4.2) sunt definite sub forma:

$$f_i(\mathbf{v}) = \varphi(\mathbf{v},l_i) - y_i,\ i = 1...m, \tag{4.4.3}$$

în care y_i sunt date obţinute experimental, susceptibile a fi afectate de erori.

Variabilele independente v_j, $j = 1...n$, sunt interpretate ca parametri ai modelului şi aceştia trebuie determinaţi astfel încât să fie asigurată o potrivire (fitting) cât mai bună a valorilor $\varphi(\mathbf{v}, l_i)$ obţinute din model cu valorile y_i obţinute experimental. În acest context $f_i(\mathbf{v})$ definită conform relaţiei (4.4.3) joacă rol de *eroare de modelare* $\rightarrow$ legătură cu modelarea.

Dacă modelul este valid $\rightarrow$ este de aşteptat ca reziduul optim $\| \mathbf{f}(\hat{\mathbf{v}}) \|$ să fie cât mai mic. De regulă este făcută presupunerea $m \geq n$ deoarece în caz contrar poate fi folosit un model arbitrar ca soluţie a sistemului de *m* ecuaţii neliniare cu *n* necunoscute:

$$\varphi(\mathbf{v}, l_i) - y_i, \; i = 1...m. \qquad (4.4.4)$$

Parametrul independent l, care ia valorile l_i, $i = 1...m$, poate fi asimilat de regulă variabilei independente timp. Dar acest parametru poate avea şi altă semnificaţie.

Pentru rezolvarea PCMMPN poate fi utilizată oricare dintre metodele de rezolvarea numerică a problemelor de programare matematică fără restricţii. Dar este avantajoasă utilizarea unor metode dedicate PCMMPN pentru a exploata structura particulară a acestor probleme – din faptul că gradientul şi hessiana lui $\theta(\mathbf{v})$ au forme speciale.

Reţele neuronale artificiale

Reţelele neuronale artificiale (RNA) – modele matematice simplificate ale sistemului nervos central → au capacitatea de a răspunde la stimuli de intrare şi de a se adapta la mediu.

Din punct de vedere sistemic o RNA reprezintă o ***clasă de aplicaţii neliniare parametrizate corespunzător***.

Algoritmii de antrenare (învăţare, adaptare) pentru reţele neuronale artificiale sunt clasificaţi în trei categorii – legătură cu fig. 1 din (Brunton et al., 2020):
(1) de învăţare supervizată,
(2) de învăţare nesupervizată,
(3) de învăţare prin reîntărire (sau recompensă, reinforcement learning).

(1) *Algoritmii de învăţare supervizată* necesită un bloc extern ("teacher"), situat la nivelul ierarhic superior, pentru specificarea vectorului ieşirilor dorite ale reţelei. Exemple: regula delta, algoritmul back-propagation (algoritmul propagării înapoi a erorii), algoritmi de învăţare hebbiană, algoritmii stochastici şi maşina Boltzmann.

(2) *Algoritmii de învăţare nesupervizată* construiesc modele interne fără supervizare externă. Exemple: de învăţare competitivă, cei specifici reţelelor ART şi reţelelor Kohonen cu autoorganizare, unii algoritmi de învăţare hebbiană.

(3) *Algoritmii de învăţare prin reîntărire (sau prin recompensă)* funcţionează pe baza unui indicator generic de performanţă (sau de calitate) a sistemului modelat. Ca indicator de performanţă poate fi utilizată orice măsură ce poate fi prelucrată de algoritmul de învăţare. În loc de obiectivul de obţinere a unor anumite ieşiri ale reţelei pe baza ieşirilor ţintă, aceşti algoritmi urmăresc obiectivul determinării acelor ieşiri ţintă ale reţelei care pot conduce la mărirea valorii indicatorului de performanţă. Exemple: algoritmii specifici automatelor cu învăţare.

Clasificarea arhitecturilor de RNA în funcţie de ***categoria algoritmului de antrenare (învăţare, adaptare)*** – conform celor trei categorii de algoritmi de antrenare:
(1) RNA cu algoritmi de învăţare supervizată: ADALINE, Boltzmann, Cascade Correlation, LVQ, perceptron, MLFF cu BP, PNN, RBFN, Alopex (ALgorithms Of Pattern EXtraction),
(2) RNA cu algoritmi de învăţare nesupervizată: ART, Hopfield, LVQ, neocognitronul, SOFM.
(3) RNA cu algoritmi de învăţare prin reîntărire.

Clasificarea RNA în funcţie de ***mecanismul intern al algoritmului de antrenare (învăţare, adaptare)***:

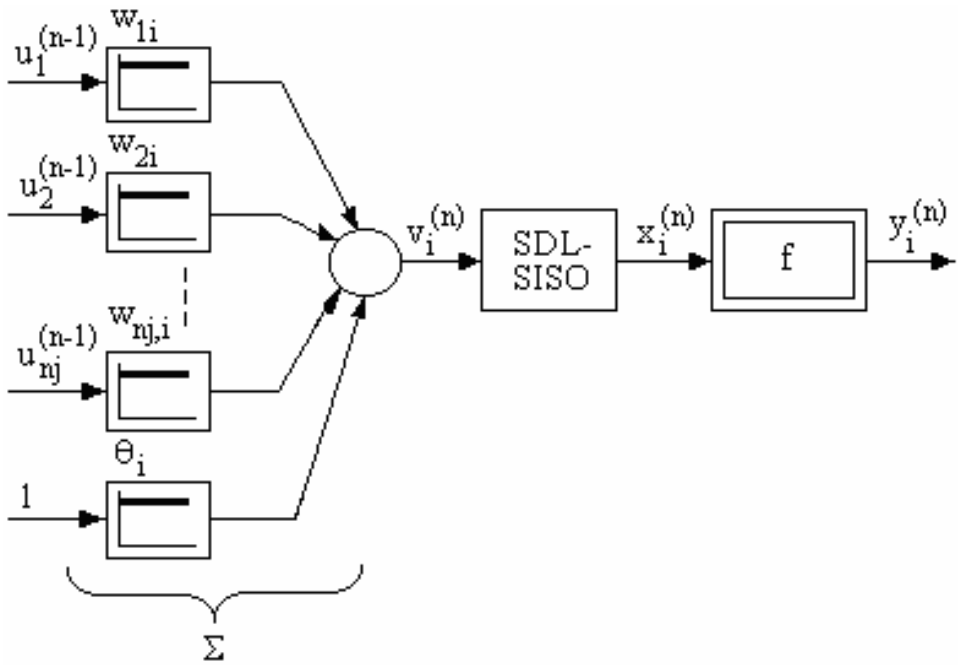
- RNA cu algoritmi de învăţare cu corecţia erorii: ADALINE, Cascade Correlation, Hopfield, perceptron, MLFF cu BP, RBFN, Alopex,
- RNA cu algoritmi de învăţare hebbiană: BSB, Hopfield, BAM, neocognitronul,
- RNA cu algoritmi de învăţare competitivă: ART, CPN, LVQ, SOFM,
- RNA cu algoritmi de învăţare stochastică: maşina Boltzmann, maşina Cauchy, Alopex.

Clasificarea RNA în funcţie de ***direcţiile de transmitere a informaţiilor (de propagare a semnalelor) în cadrul arhitecturii reţelei***:

- reţele feedforward cu un singur strat de neuroni: ADALINE, AM (Associative Memory), Hopfield, LVQ, perceptronul, SOFM,
- reţele feedforward multistrat (cu două sau mai multe straturi de neuroni): CCN, GRNN (General Regression Neural Network), MADALINE, MLFF cu BP, neocognitronul, RBF,
- reţele recurente (au conexiuni cu reacţie (feedback) care propagă ieşirile anumitor neuroni înapoi la intrarea altora, realizând prelucrarea repetată a semnalelor): ART, BAM, BSB, maşina Boltzmann, maşina Cauchy, Hopfield.

Modelul matematic asociat unui neuron artificial

Neuronul – elementul de prelucrare de bază al unei arhiecturi de RNA.
Modelul matematic asociat unui neuron artificial – schemă bloc:



Cele trei **componente** ale sale, cu referire la neuronul i din stratul n:

- Σ – sumatorul cu ponderarea intrărilor;
- SDL-SISO – sistemul dinamic liniar monovariabil la intrare şi ieşire (Single Input-Single Output, SISO);
- f – funcţia de activare (neliniară).

Sumatorul Σ realizează suma ponderată a intrărilor:

$$v_i^{(n)} = \sum_{j=1}^{n_j} w_{ji} u_j^{(n-1)} + \theta_i, \tag{6.2.1}$$

- $v_i^{(n)}$ – efectul cumulat al stimulilor de intrare (**intrărilor**) $u_j^{(n-1)}$ ai (ale) neuronului,
- w_{ji} – **ponderile** (**coeficienţii de ponderare** ai) intrărilor (**coeficienţii sinaptici**),

- θ_i – **termenul de polarizare** (**valoarea de polarizare**, **pragul**, **bias**-ul) care influenţează o intrare constantă (de valoare 1) şi are rolul modificării focarului **funcţiei de activare** *f*.

Blocul ***SDL-SISO*** este caracterizat prin funcţia de transfer (f.d.t.) $H(\lambda)$ definită astfel (în condiţii iniţiale nule):

$$H(\lambda) = x_i^{(n)}(\lambda) \big/ v_i^{(n)}(\lambda), \qquad (6.2.2)$$

$x_i^{(n)}$ – **valoarea de activare** a neuronului, $\lambda = s$ pentru SDL-SISO cu timp continuu, $\lambda = z$ pentru SDL-SISO cu timp discret.

Dacă $H(\lambda) = 1 \rightarrow$ neuronul este **fără dinamică** (**static**). În caz contrar $\rightarrow$ neuronul este **cu dinamică**, iar SDL-SISO este de regulă un element de transfer de tip I, PT1 sau cu timp mort.

(6.2.2) $\rightarrow$ în cazul neuronilor cu dinamică, aceasta poate fi introdusă cu timp continuu sau cu timp discret.

Funcţia de activare f – neliniară *monoton nedescrescătoare*, realizează o *caracteristică statică (neliniară)*, *fără dinamică*:

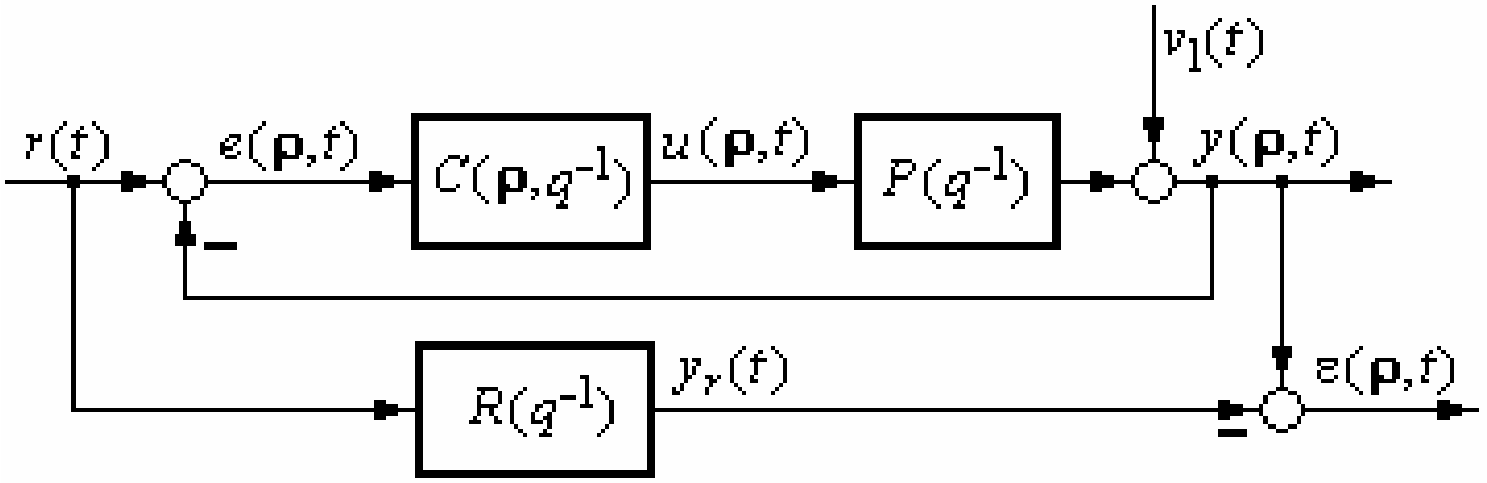
$$y_i^{(n)} = f\left(x_i^{(n)}\right), \tag{6.2.3}$$

$y_i^{(n)}$ – **ieşirea neuronului**.

Aplicaţii – legătură cu fig. 1 din (Brunton et al., 2020)

1. Tehnici iterative în acordarea parametrilor regulatoarelor automate (data-driven control model-free control)

1.1. Formulare a unor probleme de reglare automată ca probleme de urmărire a unui model de referinţă



Rezolvate ca probleme de optimizare parametrică (variabilele = parametrii de acordare ai regulatoarelor):

$$\boldsymbol{\rho}^* = \arg\min_{\boldsymbol{\rho}} J(\boldsymbol{\rho}),\ J(\boldsymbol{\rho}) = E[(1/N)\sum_{t=1}^{N} \varepsilon^2(\boldsymbol{\rho}, t)] \tag{1}$$

Procedură de căutare iterativă în mediu stohastic utilizând informaţii de gradient:

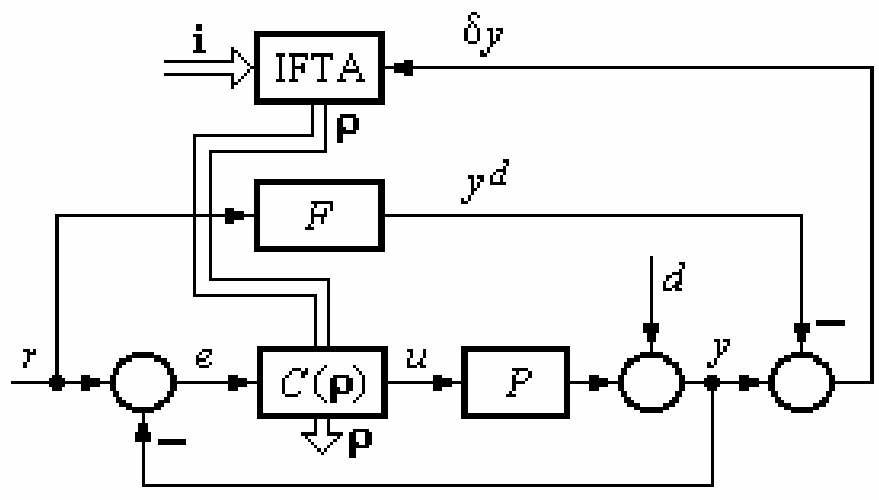
$$\boldsymbol{\rho}^{i+1} = \boldsymbol{\rho}^i - \gamma_i \mathbf{R}_i^{-1} est\{\frac{dJ}{d\boldsymbol{\rho}}(\boldsymbol{\rho}^i)\} \tag{2}$$

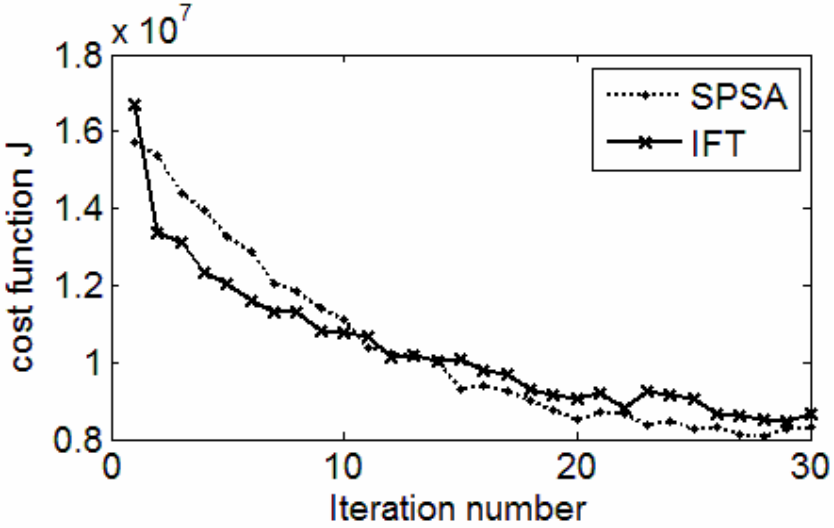
Abordare – specifică tehnicilor data-driven control: Iterative Feedback Tuning (IFT), Virtual Reference Feedback Tuning (VRFT), Correlation-based Tuning (CbT), Frequency domain Tuning (FdT), Iterative Regression Tuning (IRT), Simultaneous Perturbation Stochastic Approximation (SPSA), Pulse Response Based Control (PRBC), UnFalsified Control (UFC), Data-based Model Predictive Control (DbMPC), Model-Free Control (MFC) etc.

Informaţii de gradient:

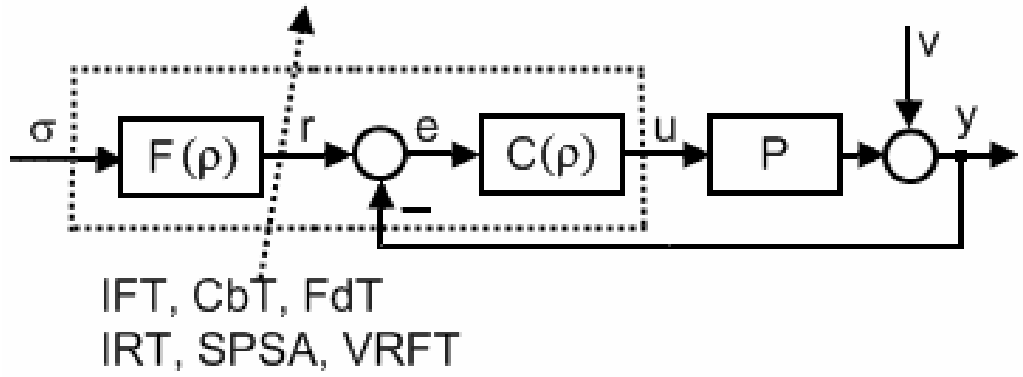
- experimente utilizând sistemul închis – cunoştinţe puţine sau chiar deloc despre modelul procesului → "model-free" ?,
- prespunere de liniaritate în majoritatea cazurilor,
- algoritm de optimizare stohastică (Robbins-Monro) – experimente afectate de zgomote.

Tehnică mixtă IFT-fuzzy pentru proiectarea şi acordarea parametrilor regulatoarelor fuzzy-PI Takagi-Sugeno. Structură & rezultate:





1.2. Tehnică de urmărire a traiectoriei referinţei utilizând experimente în formularea oferită de Iterative Learning Control (ILC)



în manieră "model-free" prin:

- (i) acordarea parametrilor regulatorului *C* sau prin
- (ii) calculul secvenţei referinţei *r*

$$y(\boldsymbol{\rho}, r) = y(\boldsymbol{\rho}_n, r) + (\partial y / \partial \boldsymbol{\rho}\,|_{\boldsymbol{\rho}=\boldsymbol{\rho}_n})(\boldsymbol{\rho} - \boldsymbol{\rho}_n) + \text{h.o.t.}$$

$$\approx y(\boldsymbol{\rho}, r_n) + (\partial y / \partial r\,|_{r=r_n})(r - r_n) + \text{h.o.t.} \qquad (3)$$

(i) – mai sus; (ii) – formulare ca problemă de optimizare:

$$J(\boldsymbol{\rho}, r) = E\{(1/N)\sum_{t=1}^{N}[y(t, \boldsymbol{\rho}, r) - y_d(t)]^2\},\ r^* = \arg\min_r J(\boldsymbol{\rho}, r) \tag{4}$$

Rezolvare – prin căutare stohastică după r:

$$r_{j+1} = r_j - \gamma_j est\{dJ/dr|_{r_j}\} \tag{5}$$

Estimarea gradientului – prin experimente pe baza ILC.

Restricţii impuse comenzii $\rightarrow$ problemă convexă pătratică rezolvată printr-un algoritm de tip Interrior Point Barrier (experimente).

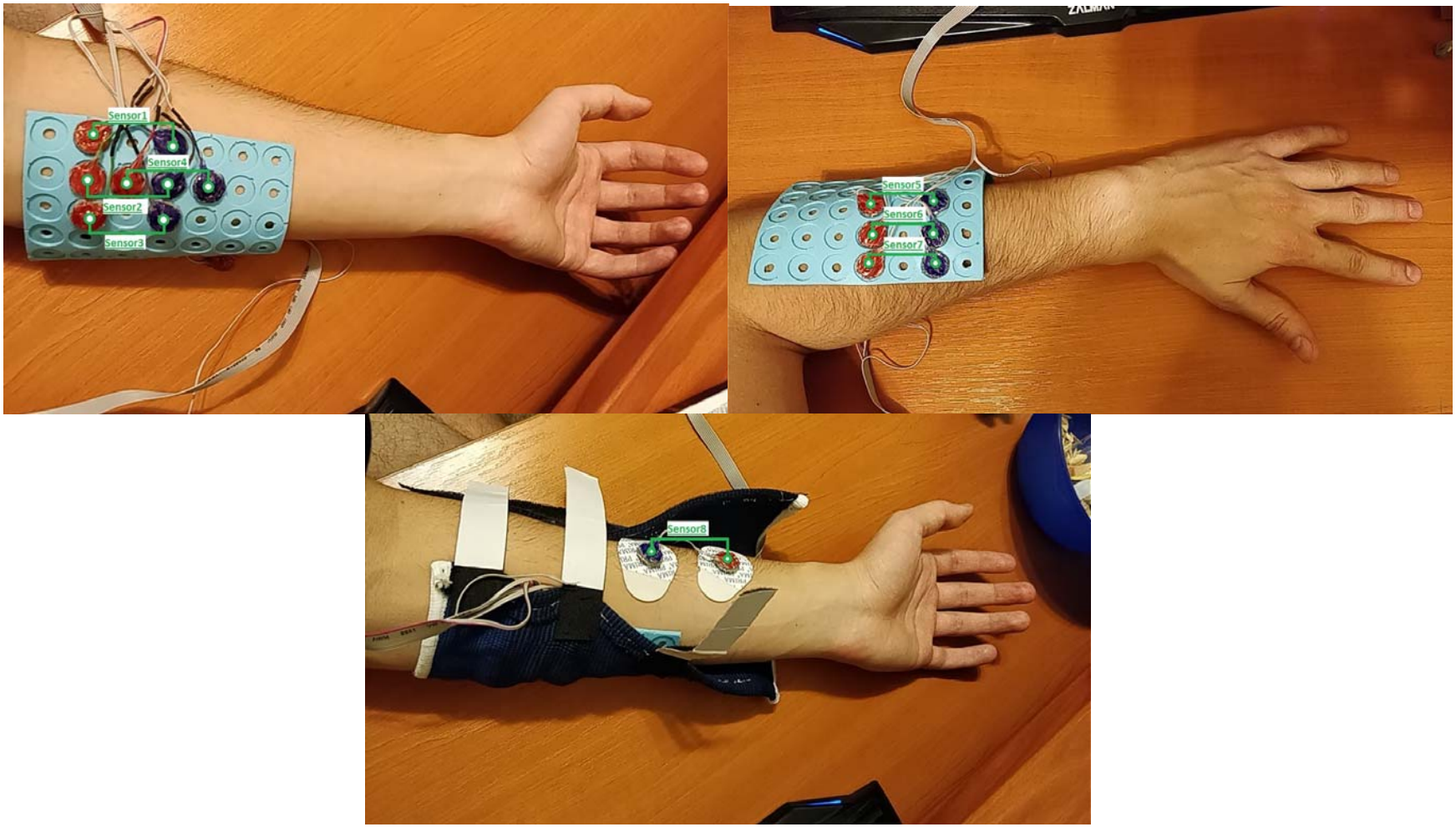
2. Modele fuzzy evolutive pentru caracterizarea dinamicii unghiurilor degetelor în funcţie de semnalele electromiografice la nivelul antebraţului

Intrări: semnale electromiografice obţinute ca ieşiri ale opt senzori amplasaţi pe antebraţ (muşchi): senzorii 1 la 4 pe flexor digitorum superficialis, senzorii 5, 6 pe extensor digitorum, sensor 7 pe extensor digiti minimi, sensor 8 pe abductor pollicis longus.

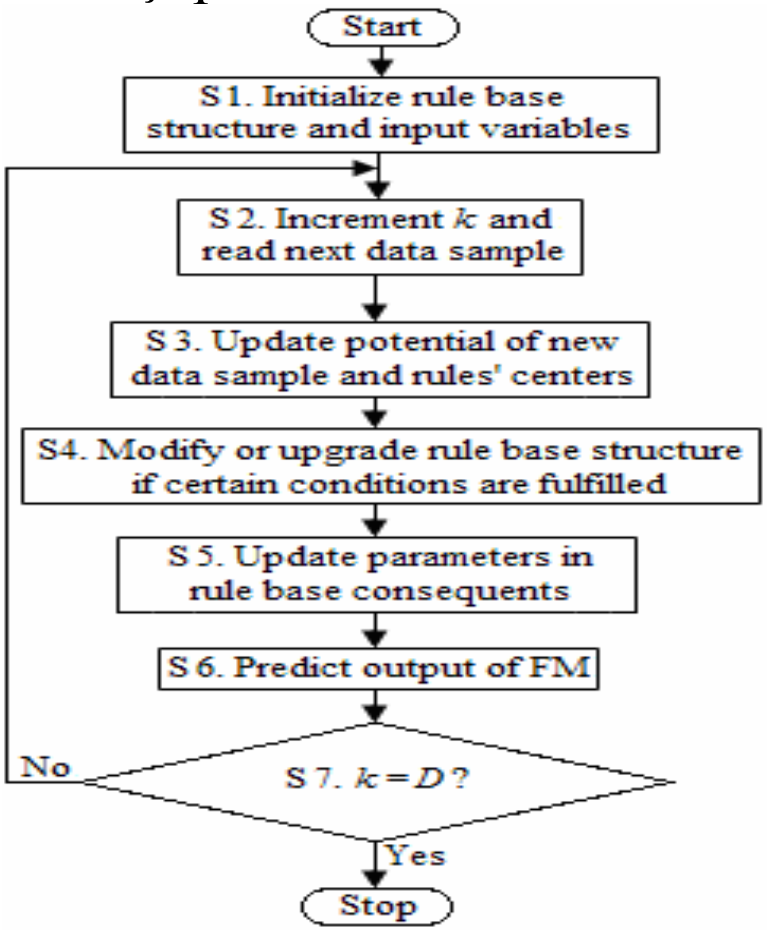
Ieşiri: unghiurile articulaţiilor carpo-metacarpiene ale celor cinci degete

Vector iniţial de intrare:

$$[z_{1,k} \;\; z_{2,k} \;\; z_{3,k} \;\; z_{4,k} \;\; z_{5,k} \;\; z_{6,k} \;\; z_{7,k} \;\; z_{8,k}]^T \qquad (6)$$


Sensor1
Sensor4
Sensor2
Sensor3
Sensor5
Sensor6
Sensor7
Sensor8

Algoritm de identificare incrementală – rezolvă iterativ o PO în sens CMMP, soluţie: structura şi parametrii modelului fuzzy:

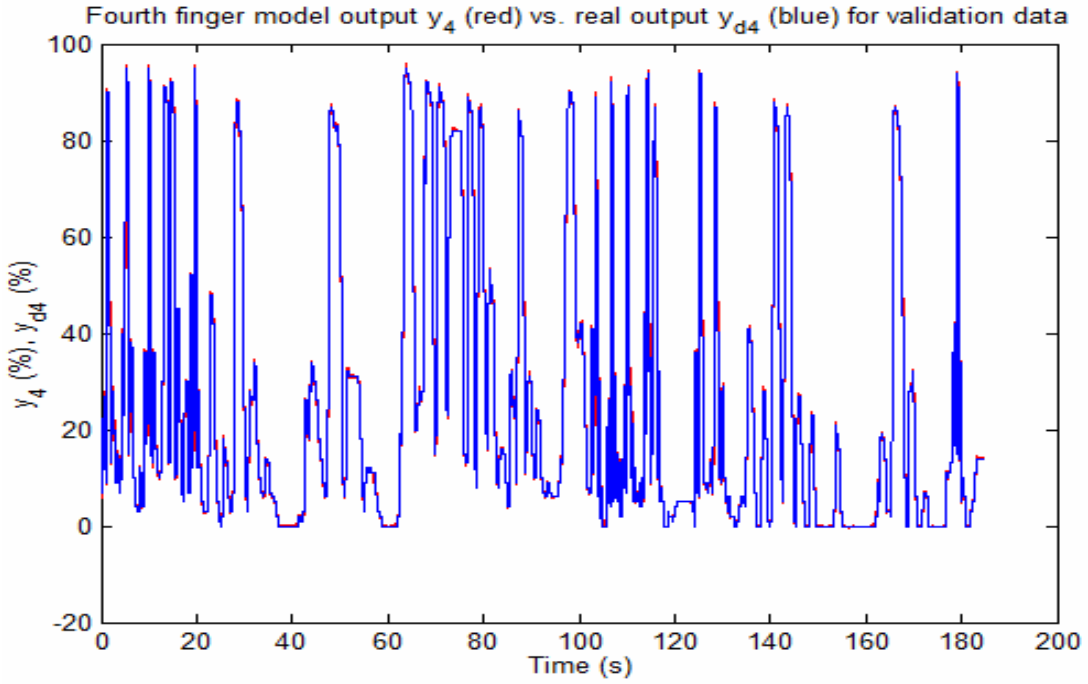


Rezultate pentru degetul 4: vector de intrare:

$$\mathbf{z}_k = [z_{1,k}\ z_{2,k}\ z_{3,k}\ z_{4,k}\ z_{5,k}\ z_{6,k}\ z_{7,k}\ z_{8,k}\ y_{1,k-1}$$

$$y_{2,k-1}\ y_{3,k-1}\ y_{4,k-1}\ y_{5,k-1}\ y_{4,k-2}]^T,$$

a evoluat la 35 de reguli, 1505 parametri, RMSE = 1.157 %.



3. Implementare unificată + bazată pe partiţii a unor algoritmi de clustering (grupare)

Este procesul de determinare a grupurilor (numite clustere) de instanţe similare (asemănătoare) dintr-un set de date. Este obţinută o descriere a unui mod în care sunt organizate instanţele unui set de date.

Din punct de vedere al învăţării automate, metodele de grupare fac parte din categoria metodelor de **învăţare nesupervizată**, clasele (grupele) instanţelor nefiind cunoscute.

Este urmărită împărţirea setului de date în grupuri, astfel încât:

- instanţele din acelaşi grup să fie mai asemănătoare (mai similare) una alteia;
- instanţele din grupuri diferite să fie mai deosebite (mai puţin asemănătoare, mai disimilare) una alteia.

Problema de optimizare care generează centroizii:

$$\mathbf{c}_{KM}^{*} = \arg\min_{\mathbf{c}\in\Re^{dk}} V(\mathbf{c}),\ V(\mathbf{c}) = \sum_{j=1}^{k} \sum_{\substack{i=1 \\ \mathbf{x}_i \in C_j}}^{n_{C_j}} d^2_{\mathbf{c}_j, \mathbf{x}_i} \tag{7}$$

Exemplificare pentru doi algoritmi: Fuzzy C-Means, K-Means.