Харківський національний університет радіоелектроніки

На правах рукопису

Копаліані Дар'я Сергіївна

УДК 004.032.26

Еволюційні нейро-фаззі мережі з каскадною структурою для інтелектуального аналізу данних

05.13.23 — системи та засоби штучного інтелекту

Дисертація на здобуття наукового ступеня
кандидата технічних наук

Науковий керівник
Бодянський Євгеній Володимирович,
доктор технічних наук, професор

Харків — 2015

ЗМІСТ

РОЗДІЛ 1
ГІБРИДНА КАСКАДНА НЕЙРО-ФАЗЗІ МЕРЕЖА З ОПТИМІЗАЦІЄЮ ПУЛУ НЕЙРОНІВ

Зазвичай під «навчанням» розуміють процес коригування синаптичних ваг, використовуючи певну процедуру оптимізації, що ґрунтується на пошуці екстремуму заданого критерію навчання. Якість процесу навчання може бути поліпшена шляхом коригування топології мережі разом з синаптичних вагами [23, 33]. Ця ідея лежить в основі систем обчислювального інтелекту, що еволюціонують [37, 47].

Мабуть найбільш відомою реалізацією цього підходу є каскадно-кореляційні нейронні мережі [27, 49, 55], привабливибі високою ефективністю та простотою налаштування як синаптичних вагових коефіцієнтів, так і топології мережі. Така мережа напочатку міститий лише один пул (ансамбль) нейронів, які навчаються назалежно один від іншого (перший каскад). Кожен нейрон у пулі може відмінні функції активації та метод навчання. Доки навчання триває, нейрони у пулі не взаємодіють один з одним. Після того як процесс налаштування вагових коефіцієнтів завершився для всіх нейронів пулу першого каскаду, кращий нейрон відповідно до обраного критерію навчання формує перший каскад і коефіцієнти його синаптичних ваг більше не коригуються. Далі формується другий каскад зазвичай з нейронів, подібних до нейронів першого каскаду. Різниця лише в тому, що нейрони, які навчаються в пулі другого каскаду мають додатковий вхід (і, отже, додатковий синаптичний ваговий коефіцієнт) - вихід першого каскаду. Подібно до першого каскаду, у другому каскаді залишиться лише один найбільш продуктивний нейрон і його синаптичні вагові коефіцієнти зафіксуються. Аналогічним чином нейрони третього каскаду матимуть два додаткових входи, а саме виходи першого та другого каскадів. Еволюційна мережа продовжуватиме розширяти свою ар-

хітектуру новими каскадами, доки вона не досягне бажаної якості вирішення завдання для заданого набору даних.

Автори найпопулярнішої каскадної нейронної мережі, що еволюціонує, CasCorLA, Фалман та Ліб'єр, використовували елементарні персептрони Розенблат з традиційними сигмоїдальними функціями активації і коригували синаптичні вагові коефіцієнти за допомогою QuickProp-алгоритму [27], що є модифікацією δ-правила. Оскільки вихідний сигнал таких нейронів нелінійно залежить від синапсових ваг, швидкість навчання не може бути суттєво збільшена для таких нейронів.

Для уникнинння багатоепохового навчання [11, 12, 15–17, 34, 41, 73] доцільно в якості вузлів системи використовувати такі типи нейронів, що їх виходи лінейно залежать від синаптичних ваг, що дозволить використовувати оптимальні за швидкодією методи навчання та обробляти дані в онлайн режимі.

Проте варто зазначити, що у випадку послідовного навчання системи неможливо визначити найкращий нейрон у пулі, адже при оброблянні нестаціонарних об'єктів певний нейрон може бути кращим для однієї частини тренувальної вибірки, але не для інших. Отже доцільно зберегти усі нейрони пулу та використовувати певну оптимізуючу процедуру (відповідну обраному критерію якості) задля визначення нейрона-переможзя на кожному кроці обробляння даних.

1.1. Архітектура оптимізованої каскадної нейронної мережі

Архітектура пропонованої гібридної системи з оптомізованим пулов нейронів у кожному каскаді наведена на ... Виходом такої системи (так званим «рецептивним» шаром) є векторний сигнал

$$x(k) = (x_1(k), x_2(k), \ldots, x_n((k))^T, \tag{1.1}$$

де $k = 1, 2, \ldots,$ – кількість обраців у таблиці «об'єкт - властивість» або поточний дискретний час.

Ці сигнали подаються на входи кожного нейрона в мережі $N_j^{[m]}$ ($j = 1, 2, \ldots, q$ – кількість нейронів у тренувальному пулі, $m = 1, 2, ...$ – номер каскаду) з вихідним сигналом $\hat{y}_j^{[m]}(k)$. Далі вихідні сигнали кожного каскаду $\hat{y}_j^{[m]}(k)$ надходять до «узагальнюючого» вузлу $GN^{[m]}$, який генерує поточно-оптимальний вихідний сигнал відповідного каскаду $\hat{y}^{*[m]}$. Слід зауважити, що вхідними сигналами першого каскаду є вектор $x(k)$ (що може містити опціональне порогове значення $x_0(k) \equiv 1$), другий каскад має додатковий вхід для сгенерованого першим каскадом вихідного сигналу $\hat{y}^{*[1]}(k)$, нейрони другого каскаду оброблятимуть два додаткових сигнали $\hat{y}^{*[1]}(k)$, $\hat{y}^{*[2]}(k)$, нейрони m-ого каскаду матимуть $(m-1)$ додаткових вхідних сигналів: $\hat{y}^{*[1]}(k)$, $\hat{y}^{*[2]}(k)$, $\hat{y}^{*[m-1]}(k)$. Під час тренування системи нові каскади додаються доки не буде досягнута бажана точність.

1.2. Навчання елементарних персептронів Розенблатта у каскадній оптимізованій системі

Наразі вважатимемо j-й вузол m-ого каскаду елементарним персепторном Розенблату з активаційною функцією

$$0 > \sigma_j^m\left(\gamma_j^{[m]} u_j^{[m]}\right) = \frac{1}{1 + e^{-\gamma_j^{[m]} u_j^{[m]}}} < 1 \tag{1.2}$$

де $u_j^{[m]}$ – внутрішній активаційний сингнал j-ого нейрону m-ого каскаду, $\gamma_j^{[m]}$ – параметр посилення.

У такому випадку вихідні сигнали нейронів тренувального пулу першого каскаду матимуть вигляд

$$\hat{y}_j^{[1]} = \sigma_J^{[1]}\left(\gamma_j^{[1]} \sum_{i=0}^{n} w_{ji}^{[1]} x_i\right) = \sigma_j^{[1]}\left(\gamma_j^{[1]} w_j^{[1]T} x\right), \tag{1.3}$$

де $w_{ji}^{[1]}$ – i-й ваговий коефіцієнт j-ого нейрону першого каскаду.

Вихідні сигнали другого каскаду дорівнюватимуть

$$\hat{y}_j^{[2]} = \sigma_J^{[2]}\left(\gamma_j^{[2]}\left(\sum_{i=0}^{n} w_{ji}^{[2]} x_i + w_{j,n+1}^{[2]}\hat{y}^{*[1]}\right)\right), \tag{1.4}$$

вихідні сигнали m-ого каскаду матимуть вигляд

$$\begin{aligned}\hat{y}_j^{[m]} &= \sigma_j^{[m]}\left(\gamma_j^{[m]}\left(\sum_{i=0}^{n} w_{ji}^{[m]} x_i + w_{j,n+1}^{[m]}\hat{y}^{*[1]}\right.\right.\\ &\left.\left. + w_{j,n+2}^{m}\hat{y}^{*[2]} + \cdots + w_{j,n+m-1}^{[m]}\hat{y}^{*[m-1]}\right)\right)\\ &= \sigma_j^{[m]}\left(\gamma_j^{[m]}\sum_{i=0}^{n+m-1} w_{ji}^{[m]} x_j^{[m]}\right) = \sigma_j^{[m]}\left(w_j^{[m]T} x^{[m]}\right),\end{aligned} \tag{1.5}$$

де $x^{[m]} = \left(x^T, \hat{y}^{*[1]}, \hat{y}^{*[m-1]}\right)^T$.

Таким чином, нейронна мережа з персептронами Розенблатта у якості вузлів, що містить m каскадів, залежить від $\left(m(n+2) + \sum_{p=1}^{m-1} p\right)$ параметрів, у тому числі від параметрів посилення $\gamma_j^{[p]}$, $p = 1, 2, \ldots, m$.

У якості критерію навчання можна використовувати загальноприйняту квадратичну функцію

$$\begin{aligned}E_j^{[m]} &= \frac{1}{2}\left(e_j^{[m]}(k)\right)^2\\ &= \frac{1}{2}\left(y(k) - \hat{y}_j^{[m]}(k)\right)^2\\ &= \frac{1}{2}\left(y(k) - \sigma_j^{[m]}\left(\gamma_j^{[m]} w_j^{[m]T} x^{[m]}(k)\right)\right)^2,\end{aligned} \tag{1.6}$$

де $y(k)$ шуканий сигнал.

Градієнтну оптимізацію критерію (1.6) відносно $w_j^{[m]}$ можна записати у вигляді

$$
\begin{aligned}
w_j^{[m]}(k+1) =& w_j^{[m]} + \eta_j^{[m]}(k+1) e_j^{[m]}(k+1) \gamma_j^{[m]} \hat{y}_j^{[m]}(k+1) \\
& \times \Big(1 - \hat{y}_j^{[m]}(k+1)\Big) x^{[m]}(k+1) \\
=& w_j^{[m]}(k) + \eta_j^{[m]}(k+1) e_j^{[m]}(k+1) \gamma_j^{[m]} J_j^{[m]}(k+1)
\end{aligned}
\tag{1.7}
$$

де $\eta_j^{[m](k+1)}$ – параметр швидкості навчання.

Мінімізувати критерій (1.6) відносно $\gamma_j^{[m]}$ можна за допомогою алгоритму Крушке-Мовелана [42]

$$
\begin{aligned}
\gamma_j^{[m]}(k+1) =& \gamma_j^{[m]}(k) + \eta_j^{[m]}(k+1) e_j^{[m]}(k+1) \hat{y}_j^{[m]}(k+1) \\
& \times \Big(1 - \hat{y}_j^{[m]}(k+1)\Big) u_j^{[m]}(k+1).
\end{aligned}
\tag{1.8}
$$

Поєднуючи (1.7) та (1.6) отримаємо алгоритм навчання для j-ого нейрону m-ого каскаду

$$
\begin{aligned}
\frac{w_j^{[m]}(k+1)}{\gamma_j^{[m]}(k+1)} =& \frac{w_j^{[m]}(k)}{\gamma_j^{[m]}(k+)} + \eta_j^{[m]}(k+1) e_j^{[m]}(k+1) \hat{y}_j^{[m]}(k+1) \\
& \times \Big(1 - \hat{y}_j^{[m]}(k+1)\Big) \left(\frac{\gamma_j^{[m]} x^{[m]}(k+1)}{u_j^{[m]}(k+1)} \right),
\end{aligned}
\tag{1.9}
$$

Або, вводячи нові змінні, у більш компактній формі

$$
\begin{aligned}
\tilde{w}_j^{[m]}(k+1) &= \tilde{w}_j^{[m]}(k) + \eta_j^{[m]}(k+1) e_j^{[m]}(k+1) \hat{y}_j^{[m]}(k+1) \tilde{x}^{[m]}(k+1) \\
&= \tilde{w}_j^{[m]}(k) + \eta_j^{[m]}(k+1) e_j^{[m]}(k+1) \tilde{J}_j^{[m]}(k+1).
\end{aligned}
\tag{1.10}
$$

Використовуючи регулюючий параметр (momentum term) [19, 58, 61] можна удосконалити процесс корегування синаптичних вагових коефіцієнтів під час навчання. Тоді, замість критерію (1.6) слід використовувати функцію

$$
\begin{aligned}
E_j^{[m]}(k) =& \frac{\eta}{2} \Big(e_j^{[m]}(k) \Big)^2 \\
& + \frac{1 - eta}{2} \left\| \tilde{w}_j^{[m]}(k) - \tilde{w}_j^{[m]}(k-1) \right\|^2, 0 < \eta \leq 1.
\end{aligned}
\tag{1.11}
$$

Тоді алгоритм навчання приймає вигляд

$$\begin{aligned}\tilde{w}_j^{[m]}(k+1) =& \tilde{w}_j^{[m]}(k) \\ &+ \eta_j^{[m]}(k+1)\Big(\eta e_j^{[m]}(k+1)\tilde{J}_j^{[m]}(k+1) \\ &+ (1-\eta)\Big(\tilde{w}_j^{[m]}(k) - \tilde{w}_j^{[m]}(k+1)\Big)\Big),\end{aligned} \tag{1.12}$$

що є модифікацією процедури Сільва-Альмеїда [58].

Доцільно вдосконалити алгоритм використовуючи підхід, запропонований у [13], тоді алгоритм (1.12) набуває слідкуючих та фільтруючих властивостей. Так, кінцева модифікація алгоритму приймає вигляд

$$\begin{cases}\tilde{w}_j^{[m]}(k+1) = & \tilde{w}_j^{[m]}(k) + \dfrac{\eta e_j^{[m]}(k+1)\tilde{J}_j^{[m]}(k+1)}{r_j^{[m]}(k+1)} \\ & + \dfrac{(1-\eta)\Big(\tilde{w}_j^{[m]}(k) - \tilde{w}_j^{[m]}(k-1)\Big)}{r_j^{[m]}(k+1)}, \\ r_j^{[m]}(k+1) = & r_j^{[m]}(k) + \left\|\tilde{J}_j^{[m]}(k+1)\right\|^2 - \left\|\tilde{J}_j^{[m]}(k-s)\right\|^2,\end{cases} \tag{1.13}$$

де s – розмір «плаваючого» вікна.

Цікаво, що при $s = 1$ та $\eta = 1$ отримуємо нелінійну версію загальновідомого алгоритму Качмажа-Уідроу-Хоффа [36, 68]:

$$\tilde{w}_j^{[m]}(k+1) = \tilde{w}_j^{[m]}(k) + \frac{e_j^{[m]}(k+1)\tilde{J}_j^{[m]}(k+1)}{\left\|\tilde{J}_j^{[m]}(k+1)\right\|^2}, \tag{1.14}$$

який широко використовується для навчання штучних нейронних мереж і відомий високою швидкістю збіжності.

1.3. Навчання нео-фаззі нейронів у оптимізованій каскадній нейронній мережі

Низька швидкість навчання персептронів Розенблатта у поєднанні з труднощами інтерпретації результатів (властиві всім ІНС в цілому) спонукає нас шукати альтернативні підходи до синтезу еволюційних нейронних мереж. Як зазначається у [35], нейро-фаззі системи відомі високою інтерпритуємістю і прозорістю, а також добрими апроксимаційними властивостями, та є основою гібридних систем штучного інтелекту. У [15, 41] розглядаються гібридні каскадні системи штучного інтелекту побудовані на нео-фаззі нейронах [48, 72], що дозволяє їм суттєво підвищити швидкість корегування синаптичних вагових коефіцієнтів. Нео-фаззі нейрон (NFN) – це нелінійна система, що реалізує висновування

$$\hat{y} = \sum_{i=1}^{n} f_i(x_i) \tag{1.15}$$

де x_i – i-й вхідний сигнал ($i = 1, 2, \ldots, n$),

$\hat{y}$ – вихідний сигнал нео-фаззі нейрону.

Структурними елементами нео-фаззі нейрона є нелінійні синапси NS_i, які трансформують вхідні сигнали в такий спосіб:

$$f_i(x_i) = \sum_{l=1}^{h} w_{li}\mu_{li}(x_i), \tag{1.16}$$

де w_{li} – l-й ваговий коефіцієнт i-ого нелінійного синапсу,

$l = 1, 2, \ldots, h$ – кількість синаптичних ваг, а отже і функцій належності $\mu_{li}(x_i)$ у синапсі.

Таким чином, нелінійний синапс NS_i реалізує нечітке висновування

$$\text{IF } x_i \text{ IS } X_{li} \text{ THEN THE OUTPUT IS } w_{li}, \tag{1.17}$$

де X_{li} – нечітка множина з функцією належності μ_{li},

w_{li} – сінглтон (синаптичний ваговий коефіцієнт у консиквенті).

Тобто нелінійній синапс фактично є системою висновування Такаґі-Суґено нульового порядку [35].

Запишемо вихідні сигнали для нейронів першого каскаду у наступному вигляді:

$$\begin{cases} \hat{y}_j^{[1]}(k) = \sum\limits_{i=1}^{n} f_{ji}^{[1]}(x(k)) = \sum\limits_{i=1}^{n}\sum\limits_{l=1}^{h} w_{jli}^{[1]}\mu_{jli}^{[1]}(x_i(k)), \\ \text{IF } x_i \text{ IS } X_{li} \text{ THEN THE OUTPUT IS } w_{li} \end{cases} \tag{1.18}$$

J-й нео-фаззі нейрон першого каскаду зображено на ... (згідно топології нейронної мережі, зображеної на ...)

Автори нео-фаззі нейрону [48, 72] в якості фунцій належності використовували традиційні трикутні структури, які задовольняють умови розбиття Руспіні:

$$\mu_{jli}^{[1]}(x_i) = \begin{cases} \dfrac{x_i - c_{j,l-1,i}^{[1]}}{c_{jli}^{[1]} - c_{j,l-1,i}^{[1]}} \text{ if } x_i \in \left[c_{j,l-1,i}^{[1]}, c_{jli}^{[1]}\right], \\ \dfrac{c_{j,l+1,i}^{[1]} - x_i}{c_{j,l+1,i}^{[1]} - c_{jli}^{[1]}} \text{ if } x_i \in \left[c_{jli}^{[1]}, c_{j,l+1,i}^{[1]}\right], \\ 0 \text{ otherwise}, \end{cases} \tag{1.19}$$

де $c_{jli}^{[1]}$ – довільно обрані центри параметрів функцій належності на інтервалі $[0, 1]$, зазвичай рівномірно розподілені.

Такий вибір функцій належності гарантує, що вхідний сигнал x_i активує лише два сусідні функції, а сума їх значень завжди дорівнюватиме 1:

$$\mu_{jli}^{[1]}(x_i) + \mu_{j,l+1,i}^{[1]}(x_i) = 1, \tag{1.20}$$

$$f_{jl}^{[1]}(x_i) = w_{jli}^{[1]}\mu_{jli}^{[1]}(x_i) + w_{j,l+1,i}^{[1]}\mu_{j,l+1,i}^{[1]}(x_i). \tag{1.21}$$

Аппроксимуючі властивості системи можна поліпшити використовуючи кубічні сплайни [15] замість трикутних функцій належності:

$$\mu_{jli}^{[1]}(x_i) = \begin{cases} \dfrac{1}{4}\left(2 + 3\dfrac{2x_i - c_{jli}^{[1]} - c_{j,l-1,i}^{[1]}}{c_{jli1}^{[1]} - c_{j,l-1,i}^{[1]}} - \left(\dfrac{2x_i - c_{jli}^{[1]} - c_{j,l-1,i}^{[1]}}{c_{jli}^{[1]} - c_{j,l-1,i}^{[1]}}\right)^3\right), \\ \text{if } x \in \left[c_{j,l-1,i}^{[1]}, c_{jli}^{[1]}\right], \\ \dfrac{1}{4}\left(2 - 3\dfrac{2x_i - c_{j,l+1,i}^{[1]} - c_{jli}^{[1]}}{c_{j,l+1,i}^{[1]} - c_{jli}^{[1]}} + \left(\dfrac{2x_i - c_{j,l+1,i}^{[1]} - c_{jli}^{[1]}}{c_{j,l+1,i}^{[1]} - c_{jli}^{[1]}}\right)^3\right), \\ \text{if } x \in \left[c_{jli}^{[1]}, c_{j,l+1,i}^{[1]}\right], \\ 0, \text{ otherwise}, \end{cases} \tag{1.22}$$

або B-сплайни [41]:

$$\mu_{jli}^{g[1]} = \begin{cases} \left.\begin{array}{l} 1, \text{ if } x_i \in \left[c_{jli}^{[1]}, c_{j,l+1,i}^{[1]}\right], \\ 0, \text{ otherwise} \end{array}\right\} \text{for } g = 1, \\ \dfrac{x_i - c_{jli}^{[1]}}{c_{j,l+g-1,i}^{[1]} - c_{jli}^{[1]}}\mu_{jli}^{g-1,[1]}(x_i) + \dfrac{c_{j,l+g,i}^{[1]} - x_i}{c_{j,l+g,i}^{[1]} - c_{j,l+g,i}^{[1]}}\mu_{j,l+1,i}^{g-1,[1]}(x_i), \\ \text{for } g > 1, \end{cases} \tag{1.23}$$

де $\mu_{jli}^{g[1]}(x_i)$ – l-й сплайн g-ого порядку.

Нескладно помітити, що при $g = 2$ отримуємо трикутні функції належності (1.19).

B-сплайни, як і трикутні функції належності, забезпечують розбиття Руспіні, але в загальному випадку вони можуть активувати довільне число функцій належності, за межами інтервалу $[0, 1]$, що може стати у нагоді для подальших каскадів.

Також у якості функцій належності нелінійних синапсів можна використовувати інші структури, такі як поліноміальні гармонійні функції, вейвлети, ортогональні функції, тощо. Проте не можна сказати наперед, які з функцій забезпечать кращі результати, тому ідея використання не одного нейрона, а пулу нейронів з різними функціями належності та активації виглядає доречною та перспективною.

За аналогією до (1.18) визначаймо вихідні сигнали інших каскадів. Так, для другого каскаду можемо записати вихідні сигнали у формі

$$\hat{y}_j^{[2]} = \sum_{i=1}^{n}\sum_{l=1}^{h} w_{jli}^{[2]}\mu_{jli}^{[2]}(x_i) + \sum_{l=1}^{h} w_{j,l,n+1}^{[2]}\mu_{j,l,n+1}^{[2]}\left(\hat{y}^{*[1]}\right), \tag{1.24}$$

вихідні сигнали для нейронів m-ого каскаду

$$\hat{y}_j^{[m]} = \sum_{i=1}^{n}\sum_{l=1}^{h} w_{jli}^{[m]}\mu_{jli}^{[m]}(x_i) + \sum_{p=n+1}^{n+m-1}\sum_{l=1}^{h} w_{jlp}^{[m]}\mu_{jlp}^{[m]}\left(\hat{y}^{*[p-n]}\right). \tag{1.25}$$

Таким чином, каскадна нейронна мережа з нео-фаззі нейронів, що сформована m каскадами, містить $h\left(\sum_{p=1}^{m-1} p\right)$ параметрів.

Введемо вектор функцій належності для j-ого нео-фаззі нейрону m-ого каскаду

$$\begin{aligned}\mu_j^{[m]}(k) = \Big(&\mu_{j11}^{[m]}(x_1(k)),\ldots,\mu_{jh1}^{[m]}(x_1(k)),\mu_{j12}^{[m]}(x_2(k)),\\ &\ldots,\mu_{jh2}^{[m]}(x_2(k)),\ldots,\mu_{jli}^{[m]}(x_i(k)),\ldots,\mu_{jhn}^{[m]}(x_n(k)),\\ &\ldots,\mu_{j1,n+1}^{[m]}\left(\hat{y}^{*[1]}(k)\right),\ldots,\mu_{jh,n+m-1}^{[m]}\left(\hat{y}^{*[m-1]}(k)\right)\Big)^T\end{aligned} \tag{1.26}$$

та відповідний вектор синаптичних вагових коефіцієнтів

$$w_j^{[m]} = \Big(w_{j11}^{[m]}, \ldots, w_{jh1}^{[m]}, w_{j12}^{[m]}, \ldots, w_{jh2}^{[m]}, \ldots, w_{jli}^{[m]}, \\ \ldots, w_{jhn}^{[m]}, w_{j1,n+1}^{[m]}, \ldots, w_{jh,n+m-1}^{[m]}, \Big)^T. \tag{1.27}$$

Тоді можемо компактно записати вихідні сигнали для j-ого нейрону m-ого каскаду

$$\hat{y}_j^{[m]}(k) = w_j^{[m]T} \mu_j^{[m]}(k). \tag{1.28}$$

У такому разі критерій навчання (1.6) приймає вигляд

$$E_j^{[m]}(k) = \frac{1}{2}\big(e_j^m(k)\big)^2 = \frac{1}{2}\Big(y(k) - w_j^{[m]T} \mu_j^{[m]}(k)\Big), \tag{1.29}$$

а мінімізувати його можна використавши модифікацію процедури [18] для «плаваючого» вікна

$$\begin{cases} w_j^{[m]}(k+1) = w_j^{[m]}(k) + \dfrac{e_j^{[m]}(k+1)\mu_j^{[m]}(k+1)}{r_j^{[m]}(k+1)}, \\ r_j^{[m]}(k+1) = r_j^{[m]}(k) + \left\|\mu_j^{[m]}(k+1)\right\|^2 - \left\|\mu_j^{[m]}(k-s)\right\|^2, \end{cases} \tag{1.30}$$

або для випадку, коли $s = 1$,

$$w_j^{[m]}(k+1) = w_j^{[m]}(k) + \frac{e_j^{[m]}(k+1)\mu_j^{[m]}(k+1)}{\left\|\mu_j^{[m]}(k+1)\right\|^2}, \tag{1.31}$$

що збігається з одношаговим оптимальним алгоритмом Качмаж-Уідроу-Гоффа.

Вочевидь, замість (1.30) можна скористатись іншими алгоритмами, як-от експоненційно зважений рекурентний метод найменших квадратів (EWRLSM), що використовується у DENFIS [38], ETS [3] та FLEXFIX [4, 46]. Та варто зауважити, що EWRLSM може бути нестійким при невисокому коефіцієнті забувкуватості.

При використанні критерію навчаняя з регулюючим параметром (momentum term) (1.6) замість (1.29) отримаємо остаточний метод навчання нео-фаззі нейрона

$$\begin{cases} w_j^{[m]}(k+1) = & w_j^{[m]}(k) + \dfrac{\eta e_j^{[m]}(k+1)\mu_j^{[m]}(k+1)}{r_j^{[m]}(k+1)} \\ & + \dfrac{(1-\eta)\Big(w_j^{[m]}(k) - w_j^{[m]}(k-1)\Big)}{r_j^{[m]}(k+1)}, \\ r_j^{[m]}(k+1) = & r_j^{[m]}(k) + \Big\|\mu_j^{[m]}(k+1)|\Big\|^2 - \Big\|\mu_j^{[m]}(k-s)\Big\|^2. \end{cases} \tag{1.32}$$

Варто наголосити, що оскільки вихідні сигнали нео-фаззі нейрону лінійно залежать від його синаптичних вагових коефіцієнтів, можна використовувати будь-які методи адаптивної лініної ідентифікації [45] (наприклад, рекурсивний метод найменших квадратів другого порядку, робастні методи, методи, що ігнорують застарілі данні, тощо), що дозволяє обробляти нестаціонарні сигнали в онлайн режимі.

1.4. Розширенні нео-фаззі нейрони в якості елементів гібридної каскадної мережі, що еволюціонує

Як зазначалося вище, розглядаючи нелінійний синапс нео-фаззі нейрону з позицій нечіткої логіки, нескладно бачити, що він є вельми схожим на шар фазифікування таких нейро-фазі систем як мережі Такаґі-Суґено-Канґа, Дженґа, Ванґа-Менделя, і, фактично реалізує нечітке висновування Такаґі-Суґено нульового порядку [65, 66]. Та задля поліпшення апроксимую-

чих властивостей таких систем видається доцільним запропонувати удосконалений нелінійний синапс, такий, що реалізує нечітке висновування довільного порядку, далі «розширений нелінійний синапс» (ENS), та зсинтезувати «розширений нео-фаззі нейрон» (ENFN), що містить такі структури замість традиційних нелінійних синапсів NS_i. Архітектури розширеного нелінійного синапсу та розширенного нео-фазі нейрону наведено на ... відповідно.

Вводячі нові змінні

$$\phi_{li}(x_i) = \mu_{li}(x_i)\big(w_{li}^0 + w_{li}^1 x_i + w_{li}^2 x_i^2 + \cdots + w_{li}^p x_i^p\big), \tag{1.33}$$

$$\begin{aligned} f_i(x_i) &= \sum_{l=1}^{h} \mu_{li}(x_i)\big(w_{li}^0 + w_{li}^1 x_i + w_{li}^2 x_i^2 + \cdots + w_{li}^p x_i^p\big) \\ &= w_{li}^0 \mu_{li}(x_i) + w_{li}^1 x_i \mu_{1i}(x_i) + \cdots + w_{li}^p x_i^p \mu_{1i}(x_i) \\ &+ w_{2i}^0 \mu_{2i}(x_i) + \cdots + w_{2i}^p x_i^p \mu_{2i}(x_i) + \cdots + w_{hi}^p x_i^p \mu_{hi}(x_i), \end{aligned} \tag{1.34}$$

$$w_i = \big(w_{1i}^0, w_{1i}^1, \ldots, w_{1i}^p, w_{2i}^0, \ldots, w_{2i}^p, \ldots, w_{hi}^p\big)^T, \tag{1.35}$$

$$\begin{aligned} \tilde{\mu}_i(x_i) = \Big(&\mu_{1i}(x_i), x_i(\mu_{1i}(x_i), \ldots, x_i^p(\mu_{1i}(x_i), \\ &\mu_{2i}(x_i), \ldots, x_i^p \mu_{2i}(x_i), \ldots, x_i^p \mu_{hi}(x_i)\Big)^T \end{aligned} \tag{1.36}$$

можна представити вихідні сигнали розширенного нео-фаззі нейрона у вигляді

$$f_i(x_i) = w_i^T \tilde{\mu}_i(x_i), \tag{1.37}$$

$$\begin{aligned} \hat{y} &= \sum_{i=1}^{n} f_i(x_i) \\ &= \sum_{i=1}^{n} w_i^T \tilde{\mu}(x_i) \\ &= \tilde{w}^T \tilde{\mu}(x), \end{aligned} \tag{1.38}$$

де

$$\tilde{w}^T = \left(w_1^T, \ldots, w_i^T, \ldots, w_n^T\right)^T, \tag{1.39}$$

$$\tilde{\mu}(x) = \left(\tilde{\mu}_1^T(x_1), \ldots, \tilde{\mu}_i^T(x_i), \ldots, \tilde{\mu}_n^T(x_n)\right)^T, \tag{1.40}$$

Таким чином, ENFN містить $(p+1)hn$ вагових коефціцєнтів та реалізує нечітке висновування Такаґі-Суґено p-ого порядку, а висновування, що його реалізує кожний розширений нелінійний синапс ENS_i можна записати у формі

$$\begin{gathered}\text{IF } x_i \text{ IS } X_{li} \text{ THEN THE OUTPUT IS} \\ w_{li}^0 + w_{li}^1 x_i + \cdots + w_{li}^p x_p,\ l = 1, 2, \ldots, h,\end{gathered} \tag{1.41}$$

що збігається з нечітким висновуванням Такаґі-Суґено p-ого порядку.

Коли подати векторний сигнал $x(k)$ на вхід $ENFN$ першого каскаду, на виході отримуємо скалярне значення

$$\hat{y}^{[1]}(k) = \tilde{w}^{[1]T}(k-1)\tilde{\mu}^{[1]}(x(k)), \tag{1.42}$$

що відрізняється від виразу (1.28) для звичайних NFN тим, що містить у $p+1$ більше параметрів, що корегуються.

Вочевидь, будь-які методи навчання нео-фаззі нейронів підійдуть і для розширених нео-фаззі нейронів. Так вирази (1.30) та (1.31) для j-ого нейрону m-ого каскаду приймають вигляд

$$\begin{cases}\tilde{w}_j^{[m]}(k+1) = \tilde{w}_j^{[m]}(k) + \dfrac{e_j^{[m]}(k+1)\tilde{\mu}_j^{[m]}(k+1)}{\tilde{r}_j^{[m]}(k+1)}, \\ \tilde{r}_j^{[m]}(k+1) = \tilde{r}_j^{[m]}(k) + \left\|\tilde{\mu}_j^{[m]}(k+1)\right\|^2 - \left\|\tilde{\mu}_j^{[m]}(k-s)\right\|^2\end{cases} \tag{1.43}$$

та

$$\tilde{w}_j^{[m]}(k+1) = \tilde{w}_j^{[m]}(k) + \frac{e_j^{[m]}(k+1)\tilde{\mu}_j^{[m]}(k+1)}{\left\|\tilde{\mu}_j^{[m]}(k+1)\right\|^2} \tag{1.44}$$

відповідно.

1.5. Оптимізація пулу нео-фаззі нейронів

Вихідні сигнали, згенеровані нейронами пулу кожного з каскадів, можна об'єднати у окремому вузлі, нейроні $GN^{[m]}$, з точністю $\hat{y}^{*[m]}(k)$, не меншою від точності будь-якого нейрону пулу $\hat{y}_j^{[m]}(k)$. Це завдання можна вирішити за допомогою підходу ансамлей нейронних мереж. Хоча відомі алгоритми не призначені для роботи в онлайн-режимі, варто розглянути методи адаптивного узагальнюючого прогнозування [51, 62].

Введемо вектор ухідних сигналів для m-ого каскаду:

$$\hat{y}^{[m]}(k) = \left(\hat{y}_1^{[m]}(k), \hat{y}_2^{[m]}(k), \ldots, \hat{y}_q^{[m]}(k)\right)^T; \tag{1.45}$$

тоді отпимальний вихідний сигнал, що його генерує нейрон $GN^{[m]}$ (що, власне, є адаптивним лінійним асоціатором [23, 33]), можна записати у формі

$$\hat{y}^{*[m]}(k) = \sum_{j=1}^{1} c_j^{[m]} \hat{y}_j^{[m]}(k) = c^{[m]T} \hat{y}^{[m]}(k) \tag{1.46}$$

з обмеженнями на «незміщенність» unbiasedness - незсувність?

$$\sum_{j=1}^{q} c_j^{[m]} = E^T c^{[m]} = 1, \tag{1.47}$$

де $c^{[m]} = \left(c_1^{[m]}, c_2^{[m]}, \ldots, c_q^{[m]}\right)^T$ та $E = (1, 1, \ldots, 1)^T$ – $(q \times 1)$-вектори.
Введемо критерій навчання на «плаваючому» вікні

$$\begin{aligned} E^{[m]}(k) =& \frac{1}{2} \sum_{\tau=k-s+1}^{k} \left(y(\tau) - \hat{y}^{*[m]}(\tau)\right)^2 \\ =& \frac{1}{2} \sum_{\tau=k-s+1}^{k} \left(y(\tau) - c^{[m]T} \hat{y}^{[m]}(\tau)\right)^2, \end{aligned} \tag{1.48}$$

зважаючи на обмеженяя (1.47), функція Лаґранжа матиме вигляд

$$L^{[m]}(k) = E^{[m]}(k) - \lambda\left(1 - E^T c^{[m]}\right), \tag{1.49}$$

де λ – невизначений Лаґранжів множник.
Мінімізуючи (1.49) відносно $c^{[m]}$, отримуємо

$$\begin{cases} \hat{y}^{*[m]}(k+1) = \dfrac{\hat{y}^{[m]T}(k+1) P^{[m]}(k+1) E}{E^T P^{[m]}(k+1) E}, \\ P^{[m]}(k+1) = \left(\displaystyle\sum_{\tau=k-s+2}^{k+1} \hat{y}^{[m]}(\tau) \hat{y}^{[m]T}(\tau) \right)^{-1} \end{cases} \tag{1.50}$$

або у рекурентній формі

$$\begin{cases} \tilde{P}^{[m]}(k+1) = P^{[m]}(k) - \dfrac{P^{[m]}(k) \hat{y}^{[m]}(k+1) \hat{y}^{[m]T}(k+1) P^{[m]}(k)}{1 + \hat{y}^{[m]T}(k+1) P^{[m]}(k) \hat{y}^{[m]}(k+1)}, \\ P^{[m]}(k+1) = \tilde{P}^{[m]}(k+1) + \\ \qquad \dfrac{\tilde{P}^{[m]}(k+1) \hat{y}(k-s+1) \hat{y}^{[m]T}(k-s+1) \tilde{P}^{[m]}(k+1)}{1 - \hat{y}^{[m]T}(k-s+1) \tilde{P}^{[m]}(k+1) \hat{y}^{[m]}(k-s+1)}, \\ \hat{y}^{*[m]}(k+1) = \dfrac{\hat{y}^{[m]T}(k+1) P^{[m]}(k+1) E}{E^T P^{[m]}(k+1) E}. \end{cases} \tag{1.51}$$

У випадку, коли $s = 1$ (1.50) та (1.51) приймають вельми простий вигляд:

$$\begin{aligned}\hat{y}^{[m]}(k+1) &= \frac{\hat{y}^{[m]T}(k+1)\hat{y}^{[m]}(k+1)}{E^T\hat{y}^{[m]}(k+1)} \\ &= \frac{\left\|\hat{y}^{[m]}(k+1)\right\|^2}{E^T\hat{y}^{[m]}(k+1)} \\ &= \frac{\sum\limits_{j=1}^{q}\left(\hat{y}^{[m]}(k+1)\right)^2}{\sum\limits_{j=1}^{q}\hat{y}^{[m]}(k+1)}.\end{aligned} \tag{1.52}$$

Важливо зазначити, що навчання як нео-фаззі нейронів, так і нейронів-узалгальнювачів можна організовувати в онлайн-режимі. Таким чином, вагові коефіцієнти нейронів попередніх каскадів (на відміну від CasCorLA) можна не замороджувати, а постійно корегувати. Так само, число каскадів не має бути фіксованим і може змінюватись у часі, що відрізняє пропоновану нейронної мережі від інших відомих каскадних систем.

РОЗДІЛ 2
БАГАТОВИМІРНА КАСКАДНА НЕЙРО-МЕРЕЖА, ЩО ЕВОЛЮЦІОНУЄ

Задача прогнозування багатовимірних часових рядів доволі часто виникає у багатьох технічних, медико-біологічних та інших дослідженнях, де якість прийнятих рішень істотно залежить від точності синтезованих прогнозів. У багатьох реальних задачах часові ряди характеризуються високим рівнем нелінійності та нестаціонарності своїх параметрів, наявністю аномальних викидів. Зрозуміло, що традиційні методи аналізу часових рядів, засновані на регресійному, кореляційному та інших подібних підходах, що мають на меті апріорну наявність доволі великої вибірки спостережень, є неефективними. Альтернативою традиційним статистичним методам може слугувати математичний апарат обчислювального інтелекту, а також штучні нейронні мережі [1, 2] та нейро-фаззі-системи [3], завдяки своїм універсальним апроксимувальним властивостям. Водночас з апроксимувальних властивостей зовсім не витікають екстраполюючі, оскільки врахування давньої передісторії для побудови прогнозувальної моделі може погіршити якість прогнозу. У зв'язку з цим під час оброблення нестаціонарних процесів треба відмовитися від процедур навчання, що базуються на зворотному поширенні помилок (багатошарові персептрони, рекурентні нейронні мережі, адаптивні нейромережеві системи нечіткого виведення – ANFIS) або методі найменших квадратів (радіально-базисні та функціонально пов'язані нейронні мережі) та скористатися процедурами на основі локальних критеріїв та «короткої» пам'яті типу алгоритма Качмажа-Уідроу-Хоффа. При цьому використані алгоритми навчання мусять забезпечувати не лише високу швидкодію, але й фільтруючі якості для придушення стохастичної «шумової» компоненти в оброблюваному сигналі. У зв'язку з цим синтез спеціалізованих гібридних систем

обчислювального інтелекту для розв'язання задач прогнозування істотно нестаціонарних часових рядів за умов невизначеності, що забезпечують разом з високою швидкістю навчання і фільтрацію завад, є досить цікавою та перспективною задачею.

Таким чином, цей розділ присвячено сиснтесзу багатовимірної гібридної системи обчислювального інтелекту, що здатна реалізувати нелінійне відображення $R^n \to R^g$ у режимі реального часу.

2.1. Багатовимірна еволюційна каскадна система, побудована на нео-фаззі нейронах

Для вирішеня задачі прогнозування та ідентифікації багатовимірних даних в умовах апріорної і поточної структурної та параметричної невизначеності як ніколи доречні переваги каскадно-кореляційної архітектури, адже системи з такою архітектурою успадковують всі переваги елементів, які використовуються в їх вузлах, а в процесі навчання автоматично підбирається необхідна кількість каскадів для того, щоб отримати модель адекватної складності для вирішення поставленого завдання [12]. Однак, слід зазначити, що каскадно-кореляційна мережа у формі, що її запропонували С. Фальман і К. Лебьер, є системою з одним виходом, тобто не здатна реалізувати нелінійне відображення $R^n \to R^g$. Це досить серйозне обмеження, оскільки більшість практичних завдань містять кілька вихідних параметрів. Тож пропонуймо наступні модифікаціїї до архітектури каскаодно-кореляційної мережі CasCorLA:

1. замість елементарних персептронів Розенблата використовувати нео-фаззі нейрони (доцільність такого рішення було детально показано у першому розділі),
2. зафіксувати кількість нейронів у кожному каскаді, що відтепер дорівнюватиме розмірності вихідного сигналу системи.

Тоді вихідний сигнал системи дорівнюватиме вектору, що складається з

вихідних сингланів нейронів останнього каскаду:

$$\hat{y}(k) = \left(\hat{y}_1^{*[m]}(k), \hat{y}_2^{*[m]}(k), \ldots, \hat{y}_d^{*[m]}(k)\right)^T, \tag{2.1}$$

де d - розмірність вихідного вектору даних, що їх треба спрогнозувати чи ідентифікувати.

Для кожного з нейронів у якості функцій належності можна використовувати трикутні конструкції:

$$\mu_{jli}^{[1]}(x_i) = \begin{cases} \dfrac{x_i - c_{d,l-1,i}^{[1]j}}{c_{dli}^{[1]j} - c_{d,l-1,i}^{[1]j}} \text{ якщо } x_i \in \left[c_{d,l-1,i}^{[1]j}, c_{dli}^{[1]j}\right], \\ \dfrac{c_{d,l+1,i}^{[1]j} - x_i}{c_{d,l+1,i}^{[1]j} - c_{dli}^{[1]j}} \text{ якщо } x_i \in \left[c_{dli}^{[1]j}, c_{d,l+1,i}^{[1]j}\right], \\ 0 \text{ у протилежному випадку}, \end{cases} \tag{2.2}$$

кубічні сплайни:

$$\mu_{jli}^{[1]}(x_i) = \begin{cases} \dfrac{1}{4}\left(2 + 3\dfrac{2x_i - c_{dli}^{[1]j} - c_{d,l-1,i}^{[1]j}}{c_{dli1}^{[1]j} - c_{d,l-1,i}^{[1]j}} - \left(\dfrac{2x_i - c_{dli}^{[1]j} - c_{d,l-1,i}^{[1]j}}{c_{dli}^{[1]j} - c_{d,l-1,i}^{[1]j}}\right)^3\right), \\ \text{якщо } x \in \left[c_{d,l-1,i}^{[1]j}, c_{dli}^{[1]j}\right], \\ \dfrac{1}{4}\left(2 - 3\dfrac{2x_i - c_{d,l+1,i}^{[1]j} - c_{dli}^{[1]j}}{c_{d,l+1,i}^{[1]j} - c_{dli}^{[1]j}} + \left(\dfrac{2x_i - c_{d,l+1,i}^{[1]j} - c_{dli}^{[1]j}}{c_{d,l+1,i}^{[1]j} - c_{dli}^{[1]j}}\right)^3\right), \\ \text{якщо } x \in \left[c_{dli}^{[1]j}, c_{d,l+1,i}^{[1]j}\right], \\ 0 \text{ у протилежному випадку}, \end{cases} \tag{2.3}$$

або B-сплайни:

$$\mu_{jli}^{g[1]} = \begin{cases} \left.\begin{array}{l} 1 \text{ якщо } x_i \in \left[c_{dli}^{[1]j}, c_{d,l+1,i}^{[1]j}\right], \\ 0 \text{ у протилежному випадку} \end{array}\right\} \text{якщо } g = 1, \\ \dfrac{x_i - c_{dli}^{[1]j}}{c_{d,l+g-1,i}^{[1]j} - c_{dli}^{[1]j}} \mu_{dli}^{g-1,[1]j}(x_i) + \dfrac{c_{d,l+g,i}^{[1]j} - x_i}{c_{d,l+g,i}^{[1]j} - c_{d,l+g,i}^{[1]j}} \mu_{d,l+1,i}^{g-1,[1]j}(x_i), \\ \text{якщо } g > 1, \end{cases} \tag{2.4}$$

де $\mu_{dli}^{g[1]j}(x_i)$ – l-й сплайн g-ого порядку.

Запишемо вихідний сингнал j-ого нео-фаззі нейрону d-ого виходу першого каскаду у вигляді

$$\begin{cases} \hat{y}_d^{[1]j}(k) = \sum\limits_{i=1}^{n} f_{di}^{[1]j}(x_i(k)) = \sum\limits_{i=1}^{n}\sum\limits_{l=1}^{h} w_{dli}^{[1]j} \mu_{dli}^{[1]j}(x_i(k)), \\ \text{ЯКЩО } x_i(k) \in X_{li}^{j} \text{ , ТОДІ ВИХІД } w_{dli}^{[1]j}. \end{cases} \tag{2.5}$$

вихідні сигнали нео-фаззі нейронів другого каскаду:

$$\begin{aligned} \hat{y}_d^{[2]j} = & \sum_{i=1}^{n}\sum_{l=1}^{h} w_{dli}^{[2]j} \mu_{dli}^{[2]j}(x_i) + \\ & \sum_{d=1}^{g}\sum_{l=1}^{h} w_{dl,n+1}^{[2]j} \mu_{dl,n+1}^{[2]j}\left(\hat{y}_d^{*[1]}\right) \ \forall \ d = 1, 2, \ldots, g \end{aligned} \tag{2.6}$$

вихідні сигнали m-ого каскаду:

$$\begin{aligned} \hat{y}_d^{[2]j} = & \sum_{i=1}^{n}\sum_{l=1}^{h} w_{dli}^{[2]j} \mu_{dli}^{[2]j}(x_i) + \\ & \sum_{d=1}^{g}\sum_{p=n+1}^{n+m-1}\sum_{l=1}^{h} w_{dlp}^{[m]j} \mu_{dlp}^{[m]j}\left(\hat{y}_d^{*[p-n]}\right) \ \forall \ d = 1, 2, \ldots, g \end{aligned} \tag{2.7}$$

РОЗДІЛ 3
КАСКАДНА НЕЙРО-МЕРЕЖА, ЩО ЕВОЛЮЦІОНУЄ, ДЛЯ ПОСЛІДОВНОГО НЕЧІТКОГО КЛАСТЕРУВАННЯ ПОТОКІВ ДАНИХ

У цьоому розділі описані архітектура та методи навчання пропонованої каскадної нейро-мережі для нечіткого кластерування, зокрема потоків даних; проведено аналіз існуючих систем, що еволюціонують, для кластерування даних, зокрема нечіткого, і розглянуті особливості та труднощі послідовного кластерування, та описні два підходи, переваги яких поєднує у собі пропонована система: нечітке та ієрархічне кластерування.

3.1. Труднощі та особливості окремих методів кластерування даних

Завдання кластерування (класифікації без учителя) досить часто зустрічається в багатьох додатках, пов'язаних з видобутком знань, де у режимі самонавчання необхідно розбити деякий вхідний нерозмічений масив даних на однорідні в прийнятому сенсі групи. Розглянемо деякі ієрархічні та розподільні методи кластерування, адже, як буде показано далі, пропована у цьому розділі самонавчанна системи поєднає у собі переваги обох підходів.

Розподільні методи кластерування (чи то жорсткі, чи нечіткі) можна назвати динамічними у тому плані, що належність певного образу до певного кластеру (кластерів для нечіткої модифікації) не є постійною. Нездатність методів розподільного кластерування самостійно визначити кількість кластерів у певному сенсі компенсується тим, що знання форми чи розміру кластерів може стати у нагоді на этапі вибору відповідних прототипів та насамперед типу відстані (міри схожесті) і суттєво поліпшити кінцеве розбиття вибірки. Але, варто зазначити чутливість таких методів до початкової ініці-

алізації, шуму і викидів, їх сприйнятливість до локальних мінімумів, адже вони ґрунтуються на оптимізації певного цільового критерію. Типові методи розподільного кластерування мають обчислювальну складність $\mathcal{O}(N)$ для тренувальною вибірки розміру N [26].

-вступне речення- Серед методів ієрархічного кластерування виділяють два основних типи: висхідні та спадні методи. Спадні методи працюють за принципом «зверху-вниз»: на початку всі образи належать до одного кластеру, який потім розбивається на все більш дрібні кластери. Більш поширеними є висхідні алгоритми, які на початку роботи поміщають кожен об'єкт до окремого кластеру, а потім об'єднують кластери у все більш крупні, доки усі образи не матимуть свій власний кластер. Таким чином будується система вкладених розбиттів. Результати таких алгоритмів зазвичай представляють у вигляді дерева - дендрограмми. Для обчислення відстаней між кластерами використовуються наступні відстані:

— одинарний зв'язок (відстань найближчого сусіда): відстань між двома кластерами визначається відстанню між двома найбільш близькими об'єктами (найближчими сусідами) у різних кластерах. Результуючі кластери мають тенденцію об'єднуватися в ланцюжки.

— повний зв'язок (відстань найбільш віддалених сусідів): відстані між кластерами визначаються найбільшою відстанню між будь-якими двома об'єктами різних кластерів (тобто найбільш віддаленими сусідами). Цей метод зазвичай працює дуже добре, коли об'єкти походять з окремих груп. Якщо ж кластери мають видовжену форму або їх природний тип є «ланцюжковим» цей метод непридатний.

— незважене попарне середнє: відстань між двома різними кластерами обчислюється як середня відстань між усіма парами об'єктів у них. Метод ефективний, коли об'єкти формують різні групи, проте він працює однаково добре і у випадках протяжних («ланцюжкового» типу) кластерів.

— зважене попарне середнє: метод ідентичний методу незваженого по-

парного середнього, за винятком того, що при обчисленнях розмір відповідних кластерів (тобто число об'єктів, що містяться в них) використовується у якості вагового коефіцієнту. Тому доцільно використовувати даний метод у випадку нерівних за розміром кластерів.

— Незважений центроїдний метод: У цьому методі відстань між двома кластерами визначається як відстань між їх центрами тяжкості.

— Зважений центроїдний метод (медіана): цей метод ідентичний попередньому, за винятком того, що при обчисленнях використовуються ваги для обліку різниці між розмірами кластерів. Тому, якщо є або підозрюються значні відмінності в розмірах кластерів, цей метод виявляється переважно попереднього.

Порівняно з розподільним кластеруванням, методи ієрархічного кластерування легко ідентифікують викидів, не потребують визначеної кількості кластерів та нечутливі до початкової ініціалізації чи локальних мінімумів. До недоліків варто віднести нездатність методів визначати кластери, що перекривають один інший. Крім того, ієрархічне кластерування є статичним, тобто образи віднесені до певного кластеру на ранніх стадіях не можуть бути пізніше належними іншому, що унеможливлює створення модифікацій методів для послідовного кластерування, на відміну від розподільного кластерування. Методи ієрархічного кластерування здебільшого мають обчислювальну складність принаймні $\mathcal{O}\left(N^2\right)$, що робить їх використання недоцільним для велких наборів даних.

3.1.0.1. Нечітке послідовне кластерування. Традиційний підхід до завдання кластерування припускає, що кожне спостереження належить лишу одному кластерові, в той час як більш природною видається ситуація, коли кожен вектор-спостереження оброблюваної вибірки можна віднести відразу декільком класам з різними рівнями належності. Така ситуація є предметом розгляду нечіткого кластерного аналізу [1, 8, 28, 30, 50, 71], а для його вирішення широко використовується апарат обчислювального інтелекту [9-12] і, насамперед, нейро-фаззі підхід [13]. При цьому більшість алгоритмів

нечіткої кластеризації призначені для роботи в пакетному режимі, коли усі дані, що підлягають обробці, задані апріорно. Вихідною інформацією для такої задачі є вибірка спостережень, сформована з m-мірних векторів ознак $x(1), x(2), \ldots, x(1), ;x(N)$, при цьому для зручності чисельної реалізації вихідні дані попередньо деяким чином перетворюються, наприклад, так, щоб всі спостереження належали до гіперкубу $[-1, 1]^n$ або одиничній гіперсфері $\|x(k)\|^2$.

Результатом такого кластерування є розбиття масиву вхідних даних на M кластерів з певним рівнем належності $u_J(k)$ k-ого вхідного образу $x(k)$ до J-ого кластеру ($J = 1, 2, \ldots, M$). Передбачається, що N та M, а також параметри кластерування (в першу чергу, фазифікатор) задані апріорі і не змінюються під час обробки даних. Варто зауважити, що існує широкий клас задач динамічного інтелектуального аналізу даних і потоків даних (Dynamyc Data Mining, Data Stream Mining) [2, 6, 20, 32, 44, 57, 69] у випадку, коли дані надходять у вигляді послідовного потоку в онлайн режимі. Отже, кількість вхідних образів N у цьому випадку не обмежується, а k набуває значення поточного дискретного часу.

Самоорганізовні мапи Когонена [40] добре пристосовані для вирішення завдання кластеризації в онлайн режимі. Ці нейронні мережі мають один шар латеральних з'єднань та навчаються за «переможець отримує все» або «переможець отримує більше» принципами. Самоорганізовні мапи також відомі своєю ефективністю вирішення задачі кластерування класів, що перетинаються. Тому, у зв'язку з дедалі більшою кількістю завдань кластерування потоків даних, з'явилися самонавчані нейро-фазі гібридні системи, що у деякому сенсі поєднують у собі самоорганізовні мапи Когонена (SOM) та метод нечітких c-середніх Бездека [7, 9, 22, 24, 31, 43, 53, 54, 59, 60, 63, 64]. Такі гібридні системи володіють обширною функціональності завдяки використанню спеціальних алгоритмів настройки, що ґрунтуються на процедурах оптимізації прийнятої цільової функції, але потребують попередньо заданої кількості класетрів та фіксованого значення фазифікатору.

3.2. Критерії гідності(дійсності?) нечіткого кластерування

TODO: partitionining coef? Оскільки коефіцієнт розбиття залежить лише від значень функції належності, йому властиві деякі недоліки. Коли фазифікатор прагне (наближається?) до 1, індекс дійсності буде однаковим для усіх c, коли фазифікатор наближається ∞

Індекс розбиття ентропії PE (Partition Entorpy Index) - ще один критерій дійсності нечіткого кластерування, запропонований (Bezdek, 1974a, 1981) , що залежить лише від значень фунції належності.

$$PE = -\frac{1}{N}\sum_{l=1}^{M}\sum_{i=1}^{N} u_{li}\log_a(u_{li}), \tag{3.1}$$

Індекс ентропії розбиття набуває значень у інтервалі $[0, \log_a M]$. Що ближче значення PE до 0, то жорсткіше розбиття вхідних даних. Значення PE близькі до верхньої межі вказують на відсутність будь-якої структури, притаманної набору вхідних даних, або на нездатність методу її виявити. Індекс ентропії розбиття має ті самі недоліки, що і коєфіцієнт розбиття. Оптимальній кількості кластерів M^* відповідає мінімальне значення (3.1). (Yang and Wu, 2001)

Фукуяма та Сугено запропонували індекс дійсності нечіткого кластерування, залежний як від рівнів належності так і від самих вхідних даних:

$$FS = \sum_{i=1}^{N}\sum_{l=1}^{M} u_{li}^{\beta}\Big(\|x_i - z_l\|^2 - \|z_l - z\|^2\Big), \tag{3.2}$$

де z та z_l – середнє арифметичне усієї виборки та образів віднесених до кластеру M_l відповідо. З визначення(3.2) видно, що малі значення FS говорять про компактні добре визначені кластери.

Нечітка множина i-ого образу визначається як

$$\tilde{A}_l = \sum_{i=1}^{N}\frac{u_{li}}{x_i}, l = 1, 2, \ldots, M \tag{3.3}$$

Ступінь, в якій A_l є підмножиною A_p визначається наступним чином

$$\begin{cases} S\left(\tilde{A}_l, \tilde{A}_p\right) = \dfrac{U\left(\tilde{A}_l \cap \tilde{A}_p\right)}{U\left(\tilde{A}_l\right)}, \\ U\left(\tilde{A}_j\right) = \sum\limits_{i=1}^{N} u_{ji} \end{cases} \tag{3.4}$$

Зважаючи на (3.4) можна запропонувати такі варіанти обчислення міри(?) подібності:

$$N_1\left(\tilde{A}_l, \tilde{A}_p\right) = \frac{S\left(\tilde{A}_l, \tilde{A}_p\right) + S\left(\tilde{A}_p, \tilde{A}_l\right)}{2}, \tag{3.5a}$$

$$N_2\left(\tilde{A}_l, \tilde{A}_p\right) = \min\left(S\left(\tilde{A}_l, \tilde{A}_p\right), S\left(\tilde{A}_p, \tilde{A}_l\right)\right), \tag{3.5b}$$

$$N_3\left(\tilde{A}_l, \tilde{A}_p\right) = S\left(\tilde{A}_l \cup \tilde{A}_p, \tilde{A}_p \cap \tilde{A}_l\right). \tag{3.5c}$$

Тоді індекс дійсності кластерування, що грунтується на нечіткій подібності, можна визначити як

$$FSim = \max_{1 \le l \le M} \max_{1 \le p \le M, p \ne l} N\left(\tilde{A}_l, \tilde{A}_p\right), \tag{3.6}$$

де міру нечіткої подібності $N\left(\tilde{A}_l, \tilde{A}_p\right)$ можна знайти за будь-яким виразом (3.5).

3.3. Архітектура каскадної мережі, що еволюціонує, для нечіткого кластерування

Архітекуру каскадної мережі, що еволюціонує, для нечіткого кластерування наведено на ..

До нульовго входу системи послідовно передаються дані у формі векторого сигналу $x(k) = (x_1(k), x_2(k), \ldots, x_1(k))^T$, де $k = 1, 2, K, N, N+1$, K — індекс поточного дискретного часу. Вхідні сигнали надходять до всіх вузлів системи $N_j^{[m]}$, де $j = 1, 2, K$, q - кількість вузлів у пулі-ансамблі, $m = 1, 2, K$ — номер каскаду. Вузол кожного каскаду призначений для online кластерування потоку данних і відрізняється від вузлів-сусідів використаним алгоритмом навчання або, у випадку спільного методу кластерування, параметрами алгоритму. Кількість кластерів для кожного каскаду є відомою і дорівнює

$m+1$. Елемент $PC_j^{[m]}$ дає оцінку якості кластерування кожного вузла у пулі, а елемент $PC^{*[m]}$ визначає найкращий елемент у пулі кожного каскаду. Елемент системи $XB^{[m]}$ оцінює загальную якість кластеризації пула, враховуючи прийняту кількість кластерів $m+1$. Таким чином, система розв'язує задачу кластерування нестаціонарного потоку даних в умовах невизначенності щодо кількості кластерів, а також їх вигляду і рівню взаємного перекриття. І, нарешті, вихідний вузол системи XB^*, порівнюючи якість кластеризації кожного з каскадів, виділяє найкращий результат — кількість кластерів, їх центроїди-прототипи та рівні належності кожного спостереження до кожного з сформованих центроїдів. Незважаючи на гадану громіздкість чисельна реалізація запропонованої архітектури не викликає принципових труднощів завдяки тому, що потік даних, що надоходить до системи, може оброблятися у паралельному режимі вузлами системи $N_j^{[m]}$[2, 6].

3.4. Адаптивне навчання вузлів каскадної нейро-фаззі системи, що еволюціонує

В основі алгоритмів навчання вузлів системи лежать алгоритми нечіткого кластерування, засновані на цільових функціях, такі, що вирішують задачу їх оптимізації при деяких апріорних припущеннях. Найбільш поширеним є ймовірнісний підхід, заснований на мінімізації цільової функції

$$E\left(u_{jl}^{[m]}(k),\, c_{jl}^{[m]}\right) = \sum_{k=1}^{N}\sum_{l=1}^{m+1}\left(u_{jl}^{[m]}(k)\right)^{\beta}\left\|x(k)-c_{jl}^{[m]}\right\|^2 \tag{3.7}$$

при обмеженнях

$$\sum_{l=1}^{m+1}(k)=1, \quad 0\le\sum_{k=1}^{N}u_{jl}^{[m]}(k)\le N \tag{3.8}$$

де $u_{ij}^{[m]}(k)\in[0,1]$ — рівень належності спостереження $x(k)$ до l-ого кластеру у j-ому вузлі каскаду m,

$c_{jl}^{[m]}$ — $(n \times 1)$ - вимірній вектор-центроїд l-ого кластеру у j-ому вузлі каскаду m,

$\beta > 1$ — параметр фазифікації (фазифікатор), що визначає розмитість кордонів між кластерами,

$k = \overline{1, N}$ — номер образу (N — кількість образів у вхідній виборці, що, у рамках класичного подходу Бездека, вважається незмінною та такою, що задана апріорі).

Вводячи функцію Лагранжа

$$L\left(u_{jl}^{[m]}(k),\, c_{jl}^{[m]},\, \lambda_j^{[m]}(k)\right) = \sum_{k=1}^{N}\sum_{l=1}^{m+1}\left(u_{jl}^{[m]}(k)\right)^{\beta}\left\|x(k) - c_{jl}^{[m]}\right\|^2 + \\ + \sum_{k=1}^{N}\lambda_j^{[m]}(k)\left(\sum_{l=1}^{m+1}u_{jl}^{[m]}(k) - 1\right) \tag{3.9}$$

(тут $\lambda_j^{[m]}(k)$ — невизначений множник Лагранжа) та вирішивши систему рівнянь Каруша-Куна-Таккера, нескладно отримати шукане рішення у вигляді

$$\begin{cases} u_{jl}^{[m]}(k) = \dfrac{\left(\left\|x(k) - c_{jl}^{[m]}\right\|^2\right)^{\frac{1}{1-\beta}}}{\sum\limits_{l=1}^{m+1}\left(\left\|x(k) - c_{jl}^{[m]}\right\|^2\right)^{\frac{1}{1-\beta}}}, \\ c_{jl}^{[m]} = \dfrac{\sum\limits_{k=1}^{N}\left(u_{jl}^{[m]}(k)\right)^{\beta}x(k)}{\sum\limits_{k=1}^{N}\left(u_{jl}^{[m]}(k)\right)^{\beta}}, \\ \lambda_j^{[m]}(k) = -\left(\left(\sum\limits_{l=1}^{m+1}\beta\left\|x(k) - c_{jl}^{[m]}\right\|^2\right)^{\frac{1}{1-\beta}}\right)^{\frac{1}{1-\beta}}, \end{cases} \tag{3.10}$$

що при $\beta = 2$ збігається з алгоритмом нечітких с-середніх Бездека (FCM) [29] і приймає форму

$$\begin{cases} u_{jl}^{[m]}(k) = \dfrac{\left\|x(k) - c_{jl}^{[m]}\right\|^{-2}}{\sum\limits_{l=1}^{m+1}\left\|x(k) - c_{jl}^{[m]}\right\|^{-2}}, \\ c_{jl}^{[m]} = \dfrac{\sum\limits_{k=1}^{N}\left(u_{jl}^{[m]}(k)^2 x(k)\right)}{\sum\limits_{k=1}^{N}\left(u_{jl}^{[m]}(k)\right)^2}. \end{cases} \tag{3.11}$$

Тут варто відзначити, що вибір фазифікатора $\beta = 2$ в (3.11) не дає жодних переваг порівняно з довільним значенням β у (3.10), у зв'язку з чим пропонується використовувати різні значення параметра фазифікації для кожного вузла пулу каскаду, після чого вибирати найкращий результат залежно від прийнятого критерію якості нечіткого кластерування [52, 56, 67].

Для послідовної обробки потоку даних, що надходять в online режимі, у [10, 14] були запропоновані рекурентні алгоритми, в основі яких лежить процедура нелінійного програмування Ерроу-Гурвіца-Удзави [5]. Так, пакетному алгоритмові (3.10) відповідає вираз

$$\begin{cases} u_{jl}^{[m]}(k+1) = \dfrac{\left\|x(k+1) - c_{jl}^{[m]}(k)\right\|^{\frac{1}{1-\beta}}}{\sum\limits_{l=1}^{m+1}\left\|x(k+1) - c_{jl}^{[m]}(k)\right\|^{\frac{1}{1-\beta}}}, \\ c_{jl}^{[m]}(k+1) = c_{jl}^{[m]}(k) + \eta(k+1)\left(u_{jl}^{[m]}(k+1)\right)^{\beta_j}\left(x(k+1) - c_{jl}^{[m]}(k)\right), \end{cases} \tag{3.12}$$

(тут $\eta(k+1)$ — параметр кроку навчання), що є узагальненням алгоритму навчання Чанга-Лі [21] і при $\beta = 2$ близьке до градієнтної процедури Парка-Дегера [53].

$$\begin{cases} u_{jl}^{[m]}(k+1) = \dfrac{\left\|x(k+1) - c_{jl}^{[m]}(k)\right\|^{-2}}{\sum\limits_{l=1}^{m+1}\left\|x(k+1) - c_{jl}^{[m]}(k)\right\|^{-2}} \\ c_{jl}^{[m]}(k+1) = c_{jl}^{[m]}(k) + \eta(k+1)\Big(u_{jl}^{[m]}(k+1)\Big)^2\Big(x(k+1) - c_{jl}^{[m]}(k)\Big) \end{cases} \tag{3.13}$$

Варто зауважити, що, розглянувши співвідношення (3.12) з позицій навчання Когоненової самоорганізвоної мапи (SOM) [40], можна помітити, що множник $\left(u_{jl}^{[m]}\right)^{\beta_j}$ відповідає функції сусідства в правилі навчання на основі принципу «переможцю дістається більше», маючи при цьому дзвонуватий вигляд.

Вочевидь, у випадку, коли $\beta_j = 1$ та $u_{jl}^{[m]}(k) \in 0, 1$, процедура (3.12) збігається з чітким алгоритмом c-середніх (HCM), коли ж $\beta_j = 0$, маємо стандартне правило навчання Когонена «переможцю дістається все» [40]

$$c_{jl}^{[m]}(k+1) = c_{jl}^{[m]}(k) + \eta(k+1)\Big(x(k+1) - c_{jl}^{[m]}(k)\Big), \tag{3.14}$$

запропоноване Каш'япом та Блейдоном [39] у шістдесятих роках минулого століття. Легко побачити, що процедура (3.14) оптимізує цільову функцію

$$E\Big(c_{jl}^{[m]}\Big) = \sum_{k=1}^{N}\left\|x(k) - c_{jl}^{[m]}\right\|^2, \quad \sum_{l=1}^{m+1} N_l = N, \tag{3.15}$$

мінімум якої збігається із середнім арифметичним

$$c_{jl}^{[m]} = \frac{1}{N}\sum_{k=1}^{N_l} x(k), \tag{3.16}$$

де N_l — кількість векторів, віднесених до l-го кластеру у процесі конкуренції.

Якщо записати (3.16) у рекуррентной формі, отримаємо оптимальний алгоритм самонавчання Ципкіна [25]

$$c_{jl}^{[m]}(k+1) = c_{jl}^{[m]}(k) + \frac{1}{N_l(k+1)}\Big(x(k+1) - c_{jl}^{[m]}(k)\Big), \tag{3.17}$$

де $N_l(k+1)$ — число векторів, віднесених до l-го кластеру в $k+1$-й момент реального часу, що є стандартною процедурою стохастичною апроксимації.

У загальному випадку алгоритм навчання (3.12) вузла можна розглядати як правило самонавчання нечіткої модифікації самоорганізовної мапи Когонена, архітектура якої наведена на рис

TODO:рис

Тут $N_{jl}^{[m]K}$ — стандартні нейрони Когонена, пов'язані між собою латеральними зв'язками, що налаштовуються згідно "переможцю дістається більше"правилу навчання на основі другого співвідношення (3.12). Вузли $N_{jl}^{[m]u}$ обчислюють рівні належності згідно першому співвідношенню (3.12). Вузли $N_j^{[m]}$ кожного з каскадів відрізняються тільки фазифікатором алгоритму самонавчання, а вузол кожного наступного каскаду містить додатково один нейрон Когонена і один елемент для розрахунку рівнів належності.

3.5. Керування каскадами самонавчаної нейро-фаззі системи, що еволюціонує

Якість кластерування кожого вузла системі може бути оцінена за допомогою будь-якого з індексів, що використовуються у задачах нечіткого кластерування [71]. Одим за найпростіших, та разом з тим найефективніших індексів є так званий «коефіцієнт розбиття», який, власне, є середнім квадратів рівнів належності всіх спостережень до кожного кластеру і має вигляд

$$PC_j^{[m]} = \frac{1}{N}\sum_{k=1}^{N}\sum_{l=1}^{m+1}\left(u_{jl}^{[m]}(k)\right)^2. \tag{3.18}$$

Цей коефіцієнт має ясний фізичний зміст: щокраще виражені кластери, то більше значення $PC_j^{[m]}$ (верхня межа — $PC_j^{[m]} = 1$), а його мінімум $PC_j^{[m]} = (m+1)^{-1}$ досягається, якщо дані належать усім кластерам рівномірно, що, вочевидь, є марним рішенням. Для розглянутої нами системи цей коефіцієнт зручний тим, що його легко розрахувати в online режимі

$$PC_j^{[m]}(k+1) = PC_j^{[m]}(k) + \frac{1}{k+1}\left(\sum_{l=1}^{m+1}\left(u_{jl}^{[m]}(k+1)\right)^2 - PC_j^{[m]}(k)\right). \quad (3.19)$$

Розрахунок коефіцієнту розбиття проводиться для кожного вузла системи разом з налаштуванням їх параметрів, тобто співвідношення (3.12) та (3.19) реалізуються одночасно. На кожному такті навчання вузол $PC^{*[m]}$ визначає найкращий елемент каскаду, що забезпечує максимальне значення коефіцієнта розбиття у кожний поточний момент k, при цьому не виключається ситуація, коли в різні моменти обробки інформації "переможцями"виявляться різні вузли.

Кожен з каскадів розглянутої системи відрізняється від інших числом кластерів, на які розбивається оброблюваний потік даних. Тому якщо вузли $PC_j^{[m]}$ і $PC^{*[m]}$ оцінюють якість кластеризації без урахування кількості сформованих класів, то вузли системи, позначені $XB^{[m]}$ та XB^*, оцінюють результати з урахуванням числа кластерів у кожному каскаді. Одним з таких показників є індекс Ксі-Бені [70], який для фіксованої вибірки з N спостережень може бути записаний у вигляді

$$XB_j^{[m]} = \frac{\left(\sum_{k=1}^{N}\sum_{l=1}^{m+1}\left(u_{jl}^{[m]}(k)\right)^2\left\|x(k) - c_{jl}^{[m]}\right\|^2\right)\Big/N}{\min\limits_{l\neq q}\left\|c_{jl}^{[m]} - c_{jq}^{[m]}\right\|^2} = \frac{NXB_j^{[m]}}{DXB_j^{[m]}} \quad (3.20)$$

Вираз (3.20) також можна записати у рекуррентній формі

$$XB_j^{[m]}(k+1) = \frac{NXB_j^{[m]}(k+1)}{DXB_j^{[m]}(k+1)} =$$
$$\frac{NXB_j^{[m]}(k)+\frac{1}{k+1}\left(\sum\limits_{l=1}^{m+1}\left(u_{jl}^{[m]}(k+1)\right)^2\left\|x(k+1)-c_{jl}^{[m]}(k+1)\right\|^2-NXB_j^{[m]}(k)\right)}{\min\limits_{l\neq q}\left\|c_{jl}^{[m]}(k+1)-c_{jq}^{[m]}(k+1)\right\|^2}, \tag{3.21}$$

при цьому рекурентні вирази (3.19) та (3.21) реалізуються одночасно.

Індекс Ксі-Бені є по суті співвідношенням відхилення всередині кластерів $NXB_j^{[m]}$ до величини поділу кластерів $DXB_j^{[m]}$. Оптимальному числу кластерів у каскаді відповідає мінімальне значення (3.20) та (3.21). Тому процес нарощування каскадів у системі продовжується доки значення індексу не почне збільшуватися. Цей процес контролює вузол архітектури XB^*.

Варто зауважити, що оскільки вузли кожного каскаду відрізняються тільки значенням фазифікатору, ефективність роботи кожного каскаду доцільно оцінювати за допомогою розширеного індексу Ксі-Бені EXB [71].

$$EXB_j^{[m]} = \frac{\left(\sum\limits_{k=1}^{N}\sum\limits_{l=1}^{m+1}\left(u_{jl}^{[m]}(k)\right)^{\beta[m]}\left\|x(k)-c_{jl}^{[m]}\right\|^2\right)\Big/N}{\min\limits_{l\neq q}\left\|c_{jl}^{[m]}-c_{jq}^{[m]}\right\|^2} = \frac{NEXB_j^{[m]}}{DEXB_j^{[m]}} \tag{3.22}$$

або його рекурентої форми

$$XB_j^{[m]}(k+1) = \frac{NXB_j^{[m]}(k+1)}{DXB_j^{[m]}(k+1)} =$$
$$\frac{NXB_j^{[m]}(k)+\frac{1}{k+1}\left(\sum_{l=1}^{m+1}\left(u_{jl}^{[m]}(k+1)\right)^{\beta^{[m]}}\left\|x(k+1)-c_{jl}^{[m]}(k+1)\right\|^2-NXB_j^{[m]}(k)\right)}{\min\limits_{l\neq q}\left\|c_{jl}^{[m]}(k+1)-c_{jq}^{[m]}(k+1)\right\|^2}, \tag{3.23}$$

де $\beta^{[m]}$ — фазифікатор найкращого з вузлів m-го каскаду.

Таким чином, процес еволюції пропонованої системи зумовлений максимізуванням поточного значення показника якості кластерування потоку даних, що надходять на обробку в онлайн режимі.

Висновки до розділу 3

У розділі 3 запропонована архітектура каскадної нейро-мережі, що еволюціонує, для нечіткого кластерування потоків даних а також online алгоритм її налаштування (самонавчання). Кожен вузол кожного каскаду системи вирішує завдання кластерування незалежно від інших, що дозволяє організувати паралельну обробку інформації в каскадах, тобто підвищити швидкодію цього процесу. Система не містить жолних порогових параметрів, що задаються суб'єктивно, а процес оцінювання якості її функціонування визначається шляхом відшукання оптимального значення певного індексу дійсності кластеру (їх поточна оцінка також проводиться в online режимі). Відмінною особливістю пропонованої системи є те, що вона самостійно визначає і поточне значення фазифікатору, і оптимальну кількість кластерів у кожну мить часу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Abonyi Janos, Feil Balazs. Cluster Analysis for Data Mining and System Identification. — Birkhäuser Basel, 2007. — ISBN: 3764379871.

[2] Adaptive Clustering for Multiple Evolving Streams / Bi-Ru Dai, Jen-Wei Huang, Mi-Yen Yeh, Ming-Syan Chen // IEEE Trans. Knowl. Data Eng. — 2006. — Vol. 18, no. 9. — P. 1166–1180. — URL: http://dx.doi.org/10.1109/TKDE.2006.137; http://doi.ieeecomputersociety.org/10.1109/TKDE.2006.137.

[3] Angelov Plamen P, Filev Dimitar P. An approach to online identification of Takagi-Sugeno fuzzy models // Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on. — 2004. — Vol. 34, no. 1. — P. 484–498.

[4] Angelov Plamen P., Lughofer Edwin. Data-driven evolving fuzzy systems using eTS and FLEXFIS: comparative analysis // Int. J. General Systems. — 2008. — Vol. 37, no. 1. — P. 45–67. — URL: http://dx.doi.org/10.1080/03081070701500059.

[5] Arrow Kenneth Joseph, Hurwicz Leonid, Uzawa Hirofumi. Studies in linear and non-linear programming. Stanford mathematical studies in the social sciences. — Stanford,Ca : Stanford university press, 1972. — ISBN: 0-8047-0562-3. — URL: http://opac.inria.fr/record=b1099343.

[6] Beringer Jürgen, Hüllermeier Eyke. Online clustering of parallel data streams // Data Knowl. Eng. — 2006. — Vol. 58, no. 2. — P. 180–204. — URL: http://dx.doi.org/10.1016/j.datak.2005.05.009.

[7] Beringer Jürgen, Hüllermeier Eyke. Online clustering of parallel data streams // Data Knowl. Eng. — 2006. — Vol. 58, no. 2. — P. 180–204. — URL: http://dx.doi.org/10.1016/j.datak.2005.05.009.

[8] Bezdek J.C. Pattern recognition with fuzzy objective function algorithms. — Kluwer Academic Publishers, 1981.

[9] Bodyanskiy Yevgeniy. Computational Intelligence Techniques for Data Analysis // Leipziger Informatik-Tage / Ed. by Klaus P. Jantke, Klaus-Peter Fähnrich, Wolfgang S. Wittig. — Vol. 72 of LNI. — GI, 2005. — P. 15–36. — URL: http://subs.emis.de/LNI/Proceedings/Proceedings72/article3735.html.

[10] Bodyanskiy Ye., Gorshkov Ye., Kolodyazhniy V. New recursive learning algorithms for fuzzy Kohonen clustering network // Proc. 17th Int. Workshop on Nonlinear Dynamics of Electronic Systems. — 2009. — P. 58–61.

[11] Bodyanskiy Yevgeniy, Grimm Paul, Teslenko Nataliya. Evolving cascaded neural network based on multidimensional Epanechnikov's kernels and its learning algorithm // Int. J. Information Technologies and Knowledge. — 2011. — Vol. 5, no. 1. — P. 25–30.

[12] Bodyanskiy Yevgeniy, Kharchenko Oleksandra, Vynokurova Olena. Hybrid cascade neural network based on wavelet-neuron // Information Theories and Application. — 2011. — Vol. 18, no. 4. — P. 335–343.

[13] Bodyanskiy Yevgeniy, Kokshenev Illya, Kolodyazhniy Vitaliy. An adaptive learning algorithm for a neo fuzzy neuron. // EUSFLAT Conf. — 2003. — P. 375–379.

[14] Bodyanskiy Ye., Kolodyazhniy V., Stephan A. Recursive fuzzy clustering algorithms // Proc. 10th East West Fuzzy Colloquium. — 2002. — P. 276–283.

[15] Bodyanskiy Yevgeniy, Viktorov Yevgen. The cascaded neo-fuzzy architecture using cubic-spline activation functions // Inf Theor Appl. — 2009. — Vol. 16, no. 3. — P. 245–259.

[16] Bodyanskiy Yevgeniy, Viktorov Yevgen. 110 9 – Intelligent Processing THE CASCADE NEO-FUZZY ARCHITECTURE AND ITS ONLINE LEARNING ALGORITHM. — 2013. — URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.386.7492.

[17] Bodyanskiy Yevgeniy, Viktorov Yevgen, Pliss Iryna. International Book Series "Information Science and Computing " 25 THE CASCADE GROWING NEURAL NETWORK USING

QUADRATIC NEURONS AND ITS LEARNING ALGORITHMS FOR ON-LINE INFORMATION PROCESSING. — 2013. — URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.386.9313.

[18] Bodyanskiy Ye V, Pliss IP, Solovyova T. Multistep optimal predictors of multidimensional non-stationary stochastic processes // Doklady AN USSR, series A. — 1986. — Vol. 12. — P. 47–49.

[19] Chen L H, Chang S. An adaptive learning algorithm for principal component analysis. // IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council. — 1995. — Vol. 6, no. 5. — P. 1255–63.

[20] Chen Y., Tu L. Density-based clustering for real-time stream data // Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. — 2007. — P. 133–142.

[21] Chung Fu Lai, Lee Tong. Fuzzy Competitive Learning // Neural Netw. — 1994. — Vol. 7, no. 3. — P. 539–551. — URL: http://dx.doi.org/10.1016/0893-6080(94)90111-2.

[22] Chung Fu-Lai, Lee Tong. Fuzzy competitive learning // Neural Networks. — 1994. — Vol. 7, no. 3. — P. 539–551. — URL: http://dx.doi.org/10.1016/0893-6080(94)90111-2.

[23] Cichocki A., Unbehauen R. Neural networks for optimization and signal processing // Journal of Signal Processing. — 1998. — Vol. 2. — P. 62–63.

[24] Crespo Fernando, Weber Richard. A methodology for dynamic data mining based on fuzzy clustering // Fuzzy Sets and Systems. — 2005. — Vol. 150, no. 2. — P. 267–284. — URL: http://dx.doi.org/10.1016/j.fss.2004.03.028.

[25] Cypkin Ja.Z., Kaplinskii A. I., Krasnenker A. S. Fundamentals of the theory of learning systems // Nauka. — 1970. — P. 252.

[26] Du K.L., Swamy M.N.S. Neural Networks and Statistical Learning. SpringerLink : Bücher. — Springer, 2013. — ISBN: 9781447155713. — URL: https://books.google.com.ua/books?id=wzK8BAAAQBAJ.

[27] Fahlman Scott E., Lebiere Christian. The cascade-correlation learning architecture // Advances in Neural Information Processing Systems 2. — Mor-

gan Kaufmann, 1990. — P. 524–532.

[28] Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition / Frank Höppner, Frank Klawonn, Rudolf Kruse, Thomas Runkler. — 1. Auflage edition. — John Wiley & Sons, 1999. — ISBN: 0471988642.

[29] Fuzzy models and algorithms for pattern recognition and image processing / James C. Bezdek, James Keller, Raghu Krisnapuram, Nikhil R. Pal. The Handbooks of fuzzy sets series. — New York : Springer, 2005. — ISBN: 0-387-24515-4. — URL: http://opac.inria.fr/record=b1102792.

[30] Gan Guojun, Ma Chaoqun, Wu Jianhong. Data Clustering: Theory, Algorithms, and Applications (ASA-SIAM Series on Statistics and Applied Probability). — SIAM, 2007. — ISBN: 0898716233.

[31] Han H., Qiao J. A self-organizing fuzzy neural network based on a growing-and-pruning algorithm // Fuzzy Systems, IEEE Transactions on. — 2010. — Vol. 18, no. 6. — P. 1129–1143.

[32] Han J, Kamber M. Mining Stream, Time-Series and Sequence Data // Data Mining: Concepts and Techniques. — 2006. — P. 467–489.

[33] Haykin S. Neural Networks: a Comprehensive Foundation. — New York, NY : Macmillan, 1994.

[34] Bodyanskiy Yevgeniy, Dolotov Artem, Pliss Iryna, Viktorov Yevgen. International Book Series "Information Science and Computing" 13 THE CASCADE ORTHOGONAL NEURAL NETWORK. — 2008.

[35] Jang Jyh-Shing Roger, Sun Chuen-Tsai, Mizutani Eiji. Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence. MATLAB curriculum series. — pub-PH:adr : Prentice-Hall, 1997. — P. xxvi + 614. — ISBN: 0-13-261066-3.

[36] Kaczmarz S. Approximate solution of systems of linear equations // Int. J. Control,. — 1993. — Vol. vol. 53. — P. 1269–1271.

[37] Kasabov Nikola. Evolving connectionist systems - the knowledge engineering approach (2. ed.). — Springer, 2007. — P. I–XXI, 1–457. — URL:

http://dx.doi.org/10.1007/978-1-84628-347-5.

[38] Kasabov Nikola K, Song Qun. DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction // Fuzzy Systems, IEEE Transactions on. — 2002. — Vol. 10, no. 2. — P. 144–154.

[39] Kashyap R. L., Blaydon Colin C. Estimation of probability density and distribution functions // IEEE Transactions on Information Theory. — 1968. — Vol. 14, no. 4. — P. 549–556. — URL: http://dx.doi.org/10.1109/TIT.1968.1054184.

[40] Kohonen Teuvo. Self-organizing maps. — Springer Science & Business Media, 2001. — Vol. 30.

[41] Kolodyazhniy V, Bodyanskiy Ye. Cascaded multiresolution spline-based fuzzy neural network // Proc. Int. Symp. on Evolving Intelligent Systems. — 2010. — P. 26–29.

[42] Kruschke John K, Movellan Javier R. Benefits of gain: Speeded learning and minimal hidden layers in back-propagation networks // Systems, Man and Cybernetics, IEEE Transactions on. — 1991. — Vol. 21, no. 1. — P. 273–280.

[43] Leng Gang, Prasad Girijesh, McGinnity T. Martin. An on-line algorithm for creating self-organizing fuzzy neural networks // Neural Networks. — 2004. — Vol. 17, no. 10. — P. 1477–1493. — URL: http://dx.doi.org/10.1016/j.neunet.2004.07.009.

[44] Liu Ying-Ho. Stream mining on univariate uncertain data // Appl. Intell. — 2013. — Vol. 39, no. 2. — P. 315–344. — URL: http://dx.doi.org/10.1007/s10489-012-0415-3.

[45] Ljung Lennart. System Identification: Theory for the User. — Prentice-Hall, Inc., 1999.

[46] Lughofer Edwin. FLEXFIS: A Robust Incremental Learning Approach for Evolving Takagi-Sugeno Fuzzy Models // IEEE T. Fuzzy Systems. — 2008. — Vol. 16, no. 6. — P. 1393–1410. — URL: http://dx.doi.org/10.1109/TFUZZ.2008.925908.

[47] Lughofer Edwin. Evolving Fuzzy Systems - Methodologies, Advanced

Concepts and Applications. — Springer, 2011. — Vol. 266 of Studies in Fuzziness and Soft Computing. — P. 1–410. — ISBN: 978-3-642-18086-6. — URL: http://dx.doi.org/10.1007/978-3-642-18087-3; http://dx.doi.org/10.1007/978-3-642-18087-3.

[48] Miki T., (Japan) T. Yamakawa. Analog Implementation of Neo-Fuzzy Neuron and Its On-board Learning. — WSEAS, 1999. — URL: http://www.worldses.org/online/.

[49] Nechyba Michael C., Xu Yangsheng. Cascade Neural Networks with Node-Decoupled Extended Kalman Filtering. — 1997.

[50] Oliveira Jose Valente de, Pedrycz Witold. Advances in Fuzzy Clustering and Its Applications. — New York, NY, USA : John Wiley & Sons, Inc., 2007. — ISBN: 0470027606.

[51] Otto Peter, Bodyanskiy Yevgeniy, Kolodyazhniy Vitaliy. A new learning algorithm for a forecasting neuro-fuzzy network // Integrated Computer-Aided Engineering. — 2003. — Vol. 10, no. 4. — P. 399–409. — URL: http://content.iospress.com/articles/integrated-computer-aided-engineering/ica00163.

[52] Pakhira Malay K, Bandyopadhyay Sanghamitra, Maulik Ujjwal. Validity index for crisp and fuzzy clusters // Pattern recognition. — 2004. — Vol. 37, no. 3. — P. 487–501.

[53] Park Dong C., Dagher Issam. Gradient Based Fuzzy c_means (GBFCM) Algorithm // IEEE International Conference on Neural Networks (ICNN'94). — Vol. III. — Orlando, FL : IEEE, 1994. — jun. — P. 1626–1631. — Intelligent Computing Research Lab, Department of ECE, FL.

[54] Pedrycz Witold, Rai Partab. Collaborative clustering with the use of Fuzzy C-Means and its quantification // Fuzzy Sets and Systems. — 2008. — Vol. 159, no. 18. — P. 2399–2427. — URL: http://dx.doi.org/10.1016/j.fss.2007.12.030.

[55] Prechelt L. Investigation of the CasCor Family of Learning Algorithms // Neural Networks. — 1997. — Vol. 10, no. 5. — P. 885–896.

[56] Roubens Marc. Fuzzy clustering algorithms and their cluster validity // European Journal of Operational Research. — 1982. — Vol. 10. — P. 294–301.

[57] Silva Bruno, Marques Nuno. Neural Network-based Framework for Data Stream Mining. // STAIRS. — 2012. — P. 294–305.

[58] Silva F. M., Almeida L. B. Speeding up back-propagation // Advanced Neural Computers / Ed. by R. Eckmiller. — Amsterdam : Elsevier, 1990. — P. 151–158.

[59] Smoothly distributed fuzzy C-means: a new self-organizing map / R. D. Pascual Marqui, A. D. Pascual Montano, K. Kochi, J. M. Carazo // Pattern Recognition. — 2001. — Vol. 34, no. 12. — P. 2395–2402. — URL: http://www.sciencedirect.com/science/article/B6V14-43W07HY-B/2/8a783959537578469a5357887f06327b.

[60] Tsao E. C. K., Bezdek J. C., Pal N. R. Fuzzy Kohonen clustering networks // Pattern Recognition. — 1994. — Vol. 27, no. 5. — P. 757–764. — URL: http://www.sciencedirect.com/science/article/B6V14-48MPPXD-1YX/2/cf0e26b7498b3a4f4b7698975495f7f9.

[61] Veitch A. C., Holmes G. A Modified Quickprop Algorithm // Neural Computation. — 1991. — Vol. 3, no. 3. — P. 310–311.

[62] Vorobyov Sergiy A., Cichocki Andrzej, Bodyanskiy Yevgeniy V. Adaptive Noise Cancellation For Multi-Sensory Signals. — 2001. — URL: http://citeseer.ist.psu.edu/404753.html.

[63] Vuorimaa Petri. Fuzzy Self–Organizing Map // Fuzzy Sets and Systems. — 1994. — Vol. 66. — P. 223–231.

[64] Vuorimaa P. Use of the Fuzzy Self-Organizing Map in pattern recognition // Proceedings of the Third IEEE Conference on Fuzzy Systems. IEEE World Congress on Computational Intelligence / Signal Process. Lab. , Tampere Univ. of Technol. , Finland. — Vol. 2. — New York, NY, USA : IEEE, 1994. — P. 798–801.

[65] Wang Li-Xin. Adaptive fuzzy systems and control - design and stability analysis. — Prentice Hall, 1994. — P. I–XVII, 1–232. — ISBN: 978-0-13-099631-2.

[66] Wang Li-Xin, Mendel Jerry M. Fuzzy basis functions, universal approximation, and orthogonal least-squares learning // IEEE Transactions

on Neural Networks. — 1992. — Vol. 3, no. 5. — P. 807–814. — URL: http://dx.doi.org/10.1109/72.159070.

[67] Wang W, Zhang Y. On fuzzy cluster validity indices // Fuzzy Sets and Systems. — 2007. — oct. — Vol. 158, no. 19. — P. 2095–2117.

[68] Widrow B., Hoff M. E. Adaptive Switching Circuits // Proceedings WESCON. — 1960. — P. 96–104.

[69] Wu Yi. Network Big Data: A Literature Survey on Stream Data Mining. — 2014. — Vol. 9. — P. 2427–2434. — URL: http://ojs.academypublisher.com/index.php/jsw/article/view/12381.

[70] Xie Xuanli Lisa, Beni Gerardo. A Validity Measure for Fuzzy Clustering // IEEE Trans. Pattern Anal. Mach. Intell. — 1991. — Vol. 13, no. 8. — P. 841–847. — URL: http://dx.doi.org/10.1109/34.85677.

[71] Xu Rui, Wunsch Donald C. Clustering algorithms in biomedical research: a review // Biomedical Engineering, IEEE Reviews in. — 2010. — Vol. 3. — P. 120–154.

[72] Yamakawa T., Tomoda S. A Neo Fuzzy Neuron and its Applications to System Identification and Prediction of the System Behaivour // Proc. 2nd Int. Conf. on Fuzzy Logic and Neural Networks (IIZUKA'92). — Iizuka, 1992. — P. 477–483.

[73] Ye Bodyanskiy, Vynokurova E, Teslenko N. Cascade GMDH-wavelet-neuro-fuzzy network // Proc. of the 4th Intern. Workshop on Inductive Modelling IWIM 2011. — 2011. — P. 22–30.