Центральноукраїнський національний технічний університет

Факультет будівництва, транспорту та енергетики

Кафедра ”Автоматизації виробничих процесів“

”Допущено до захисту“
Зав.кафедрою АВП
к.т.н., доцент
_____ __Дідик О.К.
”____“_____2025р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти

на тему

«Нейромережева реалізація прототипів управляючих систем побудованих по методу автономного адаптивного управління»

Neural network implementation of prototypes of control systems built using the method of autonomous adaptive control

Виконав: здобувач вищої освіти
IV курсу, групи АК-21
спеціальність 151 «Автоматизація та комп’ютерно-інтегровані технології»
_____ Кукуруза С.І.
”____“_____2025р

Керівник роботи
доцент, к.т.н
_____ Плєшков С.П.
”____“_____2025р

Рецензент
_____
”____“_____2025р

м. Кропивницький

ЗМІСТ

Вступ	2
1. Формальна модель нейрона і нейромережі	5
1.1. Основні поняття	5
1.2. Опис методу автономного адаптивного управління	9
1.3. Основні визначення	15
1.4. Алгебра образів	16
2 Моделювання середовища	18
3. Апарат формування і розпізнавання образів	23
3.1. Біологічний нейрон	23
3.2. Формальна модель нейрона	24
3.3. Завдання побудови ФРО	30
3.4. Розпізнавання просторово-часових образів	33
4 База знань	33
5 Система побудови і дослідження нейронних мереж	36
5.1. Актуальність системи	36
5.2. Загальна концепція системи	38
5.3. Конструктори мереж. Бібліотеки шаблонів	42
5.4. Організація обчислень в мережі	48
5.5. Аналізатори роботи мережі	49
5.6. Реалізація блоку оцінки стану	51
5.7. Реалізація моделі середовища	52
5.8. Приклад роботи програми	53
5.9. Перспективи розвитку СПДНМ	56
Висновки	57
Список літератури	58

ВСТУП

З огляду на сучасний рівень розвитку технологій, зокрема мікропроцесорної техніки, яка широко використовується навіть у побутових приладах, зросла потреба в інтелектуальних адаптивних системах керування, здатних функціонувати в умовах значної варіативності зовнішніх впливів. Крім того, актуальним є завдання розробки уніфікованої технології проектування таких систем.

Науково-технічний прогрес підтверджує доцільність біоінспірованих підходів, що ґрунтуються на моделюванні принципів, притаманних живим організмам. Зокрема, здатність біологічних систем з розвиненою нервовою організацією до адаптації в складних середовищах слугує основою для побудови технічних рішень за принципами біоімітації.

Серед таких підходів особливу роль відіграють штучні нейронні мережі (ШНМ), які набули широкого застосування у сфері штучного інтелекту, зокрема в задачах класифікації, розпізнавання образів та управління складними об'єктами. Основоположним принципом нейромережевого моделювання є конективізм — концепція побудови системи на основі взаємодії великої кількості простих однотипних елементів, інтегрованих у складну мережеву структуру. У такій структурі кожна вершина графа асоціюється з числовими ознаками (образами), а обчислювальний процес реалізується шляхом передачі сигналів між вершинами та виконанням елементарних математичних операцій.

Завдяки досягненням у галузі мікроелектроніки стало можливим створення нейропроцесорів (нейрочіпів), що складаються з великої кількості малопотужних обчислювальних елементів, здатних до паралельного виконання простих арифметичних операцій. Це забезпечує апаратну реалізацію нейромережевих алгоритмів, що істотно підвищує їхню ефективність і продуктивність у практичних застосуваннях.

З урахуванням зазначених властивостей, штучні нейронні мережі демонструють значний потенціал для використання в системах автоматизованого уп-

равління з високим ступенем невизначеності або нестабільністю об’єкта керування. Їх здатність до навчання, генералізації та адаптації дозволяє реалізувати моделі об'єктів, для яких традиційні аналітичні підходи є неефективними або неможливими.

Зокрема, нейромережеві моделі успішно застосовуються для реалізації інтелектуальних регуляторів, передбачення динаміки складних процесів, компенсації зовнішніх збурень і реконструкції нелінійних залежностей. Крім того, інтеграція нейромереж із традиційними методами управління (наприклад, PID- або адаптивним регулюванням) у гібридних схемах дозволяє підвищити надійність і точність систем управління в реальному часі.

Окремо варто відзначити перспективи застосування глибоких нейронних мереж та методів підкріплювального навчання в задачах автономного управління, таких як управління безпілотними транспортними засобами, адаптивна маршрутизація, інтелектуальні системи енергорозподілу, а також робототехнічні комплекси з високим ступенем автономності.

Подальші дослідження в цьому напрямі орієнтовані на підвищення ефективності алгоритмів навчання, зменшення енергоспоживання обчислювальних структур та розвиток спеціалізованих нейроморфних апаратних платформ, що імітують функціонування біологічного мозку з метою досягнення високої швидкодії й адаптивності.

З математичної точки зору, штучні нейронні мережі (ШНМ) можна розглядати як окремий клас методів статистичного моделювання. Цей клас, у свою чергу, охоплює три основні типи задач: оцінку щільності ймовірності, класифікацію та регресію.

Зокрема, в роботах [4, 5] показано, що за допомогою багатошарових мереж прямого поширення сигналу (feedforward networks), які навчаються за методом зворотного поширення помилки (backpropagation), можна ефективно реалізувати задачу оцінювання щільності ймовірності шляхом апроксимації змішаних гаусовських розподілів (Gaussian Mixture Models). У цьому контексті нейромережа виконує роль нелінійного апроксиматора щільності, а її навчання відповідає оптимі-

зації параметрів суміші гаусових функцій, що дозволяє точно відображати складні розподіли у вхідних даних.

Розроблений метод автономного адаптивного управління (ААУ). Передбачається, що система ААУ може бути повністю реалізована на нейронній мережі . У відмінності від традиційного використання НС для вирішення тільки завдань розпізнавання і формування образів, в методі ААУ погоджено розв'язуються завдання: розпізнавання і формування образів; отримання і зберігання знань (емпірично знайдених закономірних зв'язків образів і дій на об'єкт управління); оцінки якісних характеристик образів; ухвалення рішень (вибору дій).

Метод адаптивного автоматичного управління (ААУ) вирізняється тим, що передбачає використання надмірної кількості нейронів у нейронній мережі. Це необхідно для забезпечення здатності системи управління (ВУС) адаптуватися до змінних умов функціонування об'єкта управління (ОУ). Така особливість зумовлює потребу у створенні масштабних нейронних структур — для прикладу, людський мозок містить близько 10^{11} нейронів.

Нейронна мережа при цьому складається зі спеціалізованих нейронів, які є наближеними аналогами біологічних нейронів і адаптовані для виконання завдань ААУ. Зв’язки між цими нейронами формуються за певними правилами, що також орієнтовані на розв’язання задач адаптивного управління.

Через специфіку методу ААУ існуючі системи автоматизованого проєктування (САПР) і традиційні програмні середовища для моделювання нейронних мереж (зокрема, BrainMaker) виявляються малопридатними або повністю непридатними для розробки прототипів ВУС ААУ.

У зв’язку з цим метою бакалаврської роботи стало:

Розроблення інструментального засобу СПДНМ для моделювання та дослідження нейромережевих реалізацій прототипів ВУС ААУ.

Формування загальної архітектурної схеми нейромережевої реалізації таких прототипів.

1. ФОРМАЛЬНА МОДЕЛЬ НЕЙРОНА І НЕЙРОМЕРЕЖІ

1.1. Основні поняття

Поняття схеми було вперше запроваджене для формалізації обчислювальних процесів на паралельних обчислювальних системах. У цій роботі дане поняття адаптується для формального опису нейронних мереж, оскільки воно майже без змін відповідає потребам їх моделювання. Однією з ключових переваг такого підходу є високий потенціал до розпаралелювання обчислень у нейромережевих моделях.

У загальному випадку схемою вважається орієнтований ациклічний ортграф — граф, в якому допускаються ребра між вершинами з однаковими назвами (тобто з однаковими атрибутами). Вершини цього графа представляють операції, що параметризуються — операції, залежні від певного параметра t. Аргументами кожної операції є вхідні вершини, тобто ті, з яких виходять ребра, спрямовані до відповідної вершини. Кількість таких вхідних вершин визначає вхідну арність вершини. Аналогічно, вихідна арність — це кількість вершин, у які направлені вихідні ребра з даної вершини.

Розмір схеми

s(c) — це загальна кількість її вершин. Глибина схеми

d(c) визначається як максимальна довжина спрямованого шляху у графі. Порядок вершини вводиться рекурсивно: для вершин без вхідних ребер (витоків мережі) він дорівнює нулю, для решти — це максимум серед порядків їх вхідних вершин плюс один.

Такий підхід до формального опису нейронної мережі дозволяє розглядати її як композицію елементарних операцій, що виконуються над потоками даних, причому взаємозв’язки між цими операціями визначаються структурою графа. Це відкриває можливості для аналізу складності нейронної мережі, оцінки її глибини,

розміру та інших характеристик, які мають вирішальне значення під час розробки та оптимізації архітектур ВУС ААУ.

Крім того, графова модель забезпечує уніфікований підхід до моделювання різних типів нейронних мереж, як-то згорткових, рекурентних, багатошарових персептронів тощо. Завдяки представленню мережі у вигляді схеми, можна легко реалізувати механізми паралельного виконання обчислень, що є надзвичайно важливим для адаптивних систем управління в реальному часі.

З позицій практичної реалізації це дозволяє ефективно впроваджувати нейромережеві моделі в середовищах з обмеженими обчислювальними ресурсами, забезпечуючи при цьому необхідну продуктивність та швидкодію. Середовище СПДНМ, розроблене в рамках цієї роботи, побудоване саме на принципах графового моделювання та дозволяє будувати, аналізувати і тестувати прототипи нейромережевих реалізацій ВУС для задач ААУ.

Таким чином, у межах даного підходу нейронна мережа розглядається як схема, визначена у формалізованому вигляді. Суть цього формального опису полягає в тому, що мережа є сукупністю композицій параметризованих операцій, тобто функцій, які залежать від певного набору вхідних значень та параметрів.

Глибина схеми у цьому контексті інтерпретується як максимальний рівень вкладеності композицій, тобто найдовший шлях взаємозалежності між операціями від входу до виходу мережі. Це має безпосередній вплив на час виконання обчислень у мережі, що є критичним для задач адаптивного управління в режимі реального часу.

Важливою особливістю методології побудови нейронних мереж є те, що використовуються обчислювально прості операції, такі як зважена сума, булеві функції типу кон’юнкції та диз’юнкції тощо. Основна ідея полягає не в складності окремих операцій, а в їхній великій кількості та глибокій композиційній структурі. Саме така організація дозволяє досягти високого рівня виразності нейронної мережі навіть при простоті окремих її компонентів.

У межах конективістського підходу кожен нейрон розглядається як окрема функціональна одиниця, що виконує просту операцію над множиною вхідних си-

гналів. Типовим прикладом такої операції є обчислення зваженої суми входів із подальшим застосуванням функції активації, яка визначає вихідне значення нейрона залежно від рівня збудження. До часто використовуваних функцій активації належать порогова функція, сигмоїдальна, ReLU (Rectified Linear Unit) та інші.

У запропонованому підході до реалізації ВУС ААУ важливим є вибір структури нейронної мережі, яка забезпечує необхідну гнучкість і адаптивність. Така структура формується за допомогою задання зв'язків між нейронами, відповідно до орієнтованого ациклічного графа. Це дозволяє формувати як прямі (feedforward) архітектури, так і більш складні рекурентні мережі, у яких існують зворотні зв'язки.

Особливу увагу слід приділити параметрам нейронної мережі, а саме: вагам зв'язків, порогам активації нейронів та функціональним залежностям між елементами мережі. Адаптація цих параметрів у процесі навчання є ключовим аспектом функціонування систем ААУ. У більшості випадків навчання відбувається шляхом мінімізації функції помилки, що відображає відхилення виходу мережі від бажаного результату. Залежно від типу задачі використовуються різні методи оптимізації, серед яких градієнтний спуск, генетичні алгоритми, методи еволюційного навчання тощо.

На наступному етапі доцільним є впровадження цих формалізованих принципів у програмно-модельне середовище. У межах цієї бакалаврської роботи реалізовано систему СПДНМ, яка дозволяє будувати, конфігурувати та тестувати нейромережеві прототипи ВУС ААУ. Середовище надає користувачу інтерфейс для опису структури мережі у вигляді схеми, налаштування параметрів нейронів і візуалізації результатів роботи.

Цей підхід є характерним для конективістської парадигми (конективізму), в основі якої лежить уявлення про те, що інтелектуальна поведінка може виникати з великої кількості простих взаємодіючих елементів. Саме така модель особливо добре підходить для реалізації ВУС ААУ, де адаптація та узгодження з динамічними умовами середовища є ключовими вимогами. Параметр t є по суті тимчасовим параметром. Відмітимо, що тут і далі вважаємо час дискретним, хоча для фо-

рмалізації НС це не принципово. Перенумерувавши вершини схеми, можна записати загальний вид операції, що параметризується:

$$x_i(t) = f_i(x_{i_1}(t - d_{i_1 i}), x_{i_2}(t - d_{i_2 i}), ..., x_{i_{w_{in}\ i}}(t - d_{i_{w_{in}} i i}))\ ,$$

де (t) - і-а операція, що параметризується, - вхідні вершини, - синаптична затримка на ребрі $i_j \to i$. Конкретний вид функції f_i для пропонованої моделі нейрона буде представлений в розділі «Апарат ФРО».

Виходами підграфа $G(V, N)$, де V - безліч ребер, N – безліч вершин. мережі називатимемо всі ребра, входами всі ребра $i \to j : i \notin V, j \in V$.

Визначимо блок як зв'язний підграф мережі з одним виходом.

Назвемо блок $B(V, N)$ шаблоном деякого блоку $B'(V', N')$ якщо між цими блоками існує ізоморфне відображення, тобто така пара відображень
$f : V \leftrightarrow V', g : N \leftrightarrow N' : v \in V, w \in V, (v \to w) \in N, v' = f(v), w' = f(w) \Rightarrow$
$g(v \to w) = (v' \to w')$

Розбиттям мережі на блоки з шаблоном В називатимемо сукупність непересічних блоків таку, що для всіх цих блоків B є шаблоном і об'єднання всіх блоків і міжблокових ребер (є зважаючи на два різні об'єднання: безлічі вершин і безлічі ребер) є вся мережа.

Сукупність рекурсивного розбиття мережі, де $\sigma_r(B_r)$ є розбиття шаблону B_{r-1} називатимемо конструкцією мережі, а множина $\{B_1, B_2, ..., B_n\}$ шаблонами конструктора.

Таким чином, під формальною моделлю нейрона розумітимемо шаблон B_1 розбиття мережі, у якого вихід булева операція. Під нейроном розумітимемо власне блок.

На рис.1.1 представлена формальна модель перспептрона, де всі блоки $B^i{}_r$ мають один шаблон МакКаллока-Піттса.

Взагалі кажучи, стан навченості нейрона для кожної формальної моделі визначається по своєму і, неформально виражаючись, цей стан, в якому вважається, що нейрон вже «навчений» для вирішення свого завдання класифікації. Відзначимо, що процес навчання незворотній.

Говоритимемо, що мережею розпізнаний образ і, якщо після пред'явлення мережі деякого вхідного сигналу на виході і-ого нейрона з'являється 1.

Розпізнавання образу є по суті позитивна відповідь в рішенні задачі класифікації для даного нейрона.

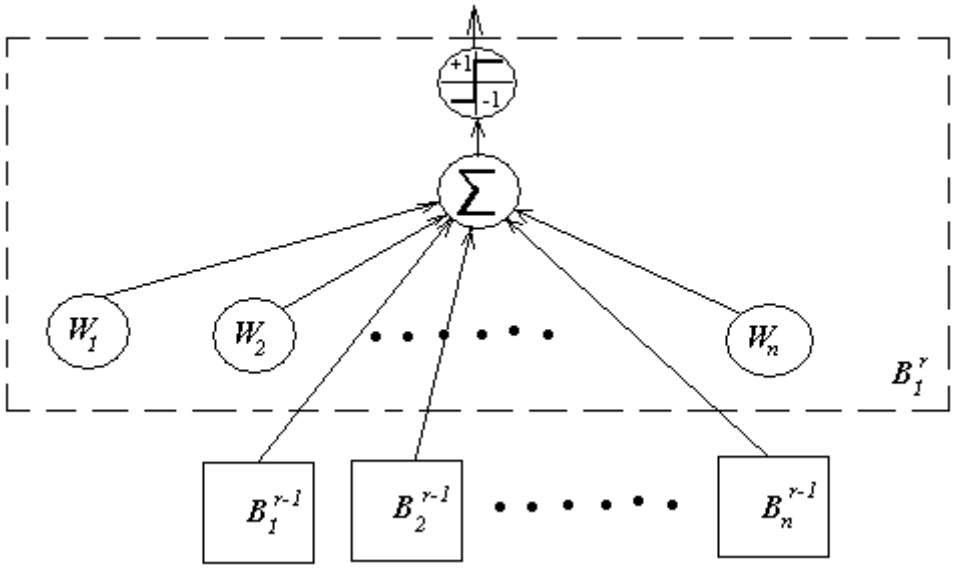


Рис.1.1. Формальна модель перспептрона

1.2. Опис методу автономного адаптивного управління.

Метод ААУ детально описаний в роботах [4,5], тому представимо тільки основні його сторони. Називатимемо системою (ВУС), що управляє, систему управління, що імітує нервову систему відповідно до методології ААУ. Під об'єктом управління (ОУ) розумітимемо організм, який несе в собі нервову систему, іншими словами, ОУ - це об'єкт, який повинен управлятися за допомогою ВУС, розташованої усередині ОУ і взаємодіючої з своїм оточенням за допомогою блоку датчиків (БД) і виконавчих органів (ВО).

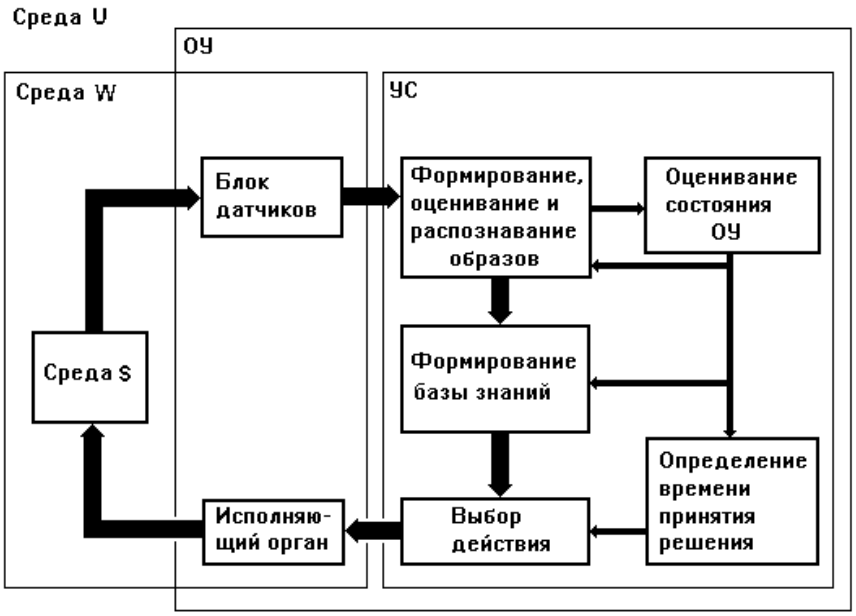


Рис. 1.2. Середовище з об'єктом управління

На рис. 1.2. представлена система, під якою розумітимемо середовище, в яке вкладений ОУ, в свою чергу що містить в собі ВУС. Як видно з малюнка, можна стверджувати, що ВУС управляє не тільки ОУ, але всією системою. Під середовищем в системі можна розуміти різні об'єднання об'єктів. Називатимемо середовищем *W* сукупність об'єктів, лежачих зовні ВУС; середовищем *S* - сукупність об'єктів, лежачих поза ОУ; середовищем *U* - всю систему.

Блок датчиків поставляє ВУС вхідну інформацію у вигляді двійкового вектора. Цей блок необхідний в реальних системах для сполучення середовища і ВУС, тому при моделюванні ВУС на ЕОМ не використовувався і ми не акцентуємо увагу на ньому в даній роботі.

Роботу блоку формування і розпізнавання образів (ФРО) можна представити наступним способом. На вхід блоку надходять дані, що описують поточний стан об’єкта управління або навколишнього середовища. Ці дані можуть мати різну природу: числові вимірювання, сигнали з датчиків, векторні уявлення вхідних характеристик тощо. Первинна обробка вхідних даних здійснюється з метою виділення ознак, які є інформативними для подальшої ідентифікації станів або ситуацій.

На етапі формування образів відбувається трансформація сирих даних у компактні векторні представлення, що фіксують характерні ознаки класів або кластерів станів. Для цього можуть застосовуватись методи зниження розмірності (наприклад, головні компоненти), кластеризація, або попередньо навчені шари нейронної мережі.

Далі сформовані образи надходять до підсистеми розпізнавання, яка здійснює класифікацію вхідних образів відповідно до заздалегідь визначених категорій або шаблонів. У цій частині блоку ФРО зазвичай реалізуються нейронні мережі або інші алгоритми машинного навчання, які дозволяють здійснювати розпізнавання навіть за умови неповноти, зашумленості або варіативності вхідних даних.

Ключовою особливістю ФРО у контексті адаптивних систем управління є його здатність до навчання та перенавчання в процесі експлуатації. Це забезпечує адаптивність до нових або змінених ситуацій, що особливо важливо для складних або динамічних середовищ функціонування об’єкта управління.

У підсумку, блок ФРО виконує функцію інтелектуального інтерпретатора вхідної інформації, переводячи її у форму, зручну для подальшої обробки в рамках ВУС ААУ. Його ефективність безпосередньо впливає на здатність всієї системи до коректного реагування на зміну умов або порушення в роботі об’єкта.

Блок ухвалення рішень (БПР), також відомий як блок вибору дії, виконує функцію формування керуючого впливу на основі аналізу поточної ситуації, цільових орієнтирів, вмісту бази знань (БЗ) та оцінки якості поточного стану, позначеної як $S(t)$.

Актуальна інформація про стан системи подається у вигляді множини образів, які були розпізнані блоком формування і розпізнавання образів (ФРО). Крім того, значення оцінки $S(t)$ відображає якість поточного функціонування об’єкта управління (ОУ) з позиції цільової функції. На основі розпізнаних образів відбувається активація відповідного підрозділу бази знань, що репрезентує релевантні знання — ті, які є дійсними в поточних умовах середовища.

Згідно з першою цільовою функцією, яка орієнтована на покращення якості стану ОУ, система здійснює вибір дії, що має максимальне сумарне значення оцінок пов’язаних із нею образів — як тих, що активуються, так і тих, що витісняються у разі її реалізації. Це дозволяє ВУС максимально ефективно адаптуватися до змін і підтримувати бажаний рівень функціонування системи.

Після вибору узагальненої дії Y (j) з БЗ, система переходить до вибору Y (j) . Цей вибір відбувається випадковим чином, що відповідає другій цільовій функції — спрямованій на збагачення знань системи через дослідницьку поведінку. Такий підхід дозволяє уникати жорсткої детермінованості в діях системи та забезпечує її здатність до навчання і самовдосконалення в умовах невизначеності або нових ситуацій.Блок оцінки стану (БОС) виробляє інтегральну оцінку якості стану ОУ *St*. Оцінка *St* використовується для розрахунку оцінки (ваги) *pi* кожного із знов сформованих образів деяким статистичним способом. У свою чергу, *St* функціонально залежить від оцінок *pi* розпізнаних образів. Є деяка безліч спочатку сформованих і оцінених образів. Оцінка *St* використовується також для розрахунку темпу ухвалення рішень.

Завдяки поєднанню двох цільових функцій — орієнтованої на оптимізацію (поліпшення поточного стану) та евристичної (отримання нових знань) — БПР забезпечує баланс між експлуатацією вже набутих знань і дослідженням нових варіантів поведінки, що є необхідним для гнучкої та адаптивної роботи в умовах нестабільного або неповного середовища.

Процедура ухвалення рішення в БПР реалізується через поетапну активацію підсистем:

Ідентифікація активної підмножини БЗ, що відповідає множині розпізнаних образів від блоку ФРО.

Обчислення цільових функцій для кожної потенційної дії на основі значень оцінок S (t) та структури знань.

Вибір узагальненої дії з максимальною сукупною корисністю.

Стохастичний вибір конкретної реалізації дії з множини варіантів, асоційованих з обраною узагальненою дією.

Цей підхід дозволяє реалізувати контекстно-залежне управління, де вибір дії прямо залежить від оцінки ситуації та знань, релевантних поточному стану об'єкта. Крім того, стохастичний компонент у виборі забезпечує варіативність поведінки, що, своєю чергою, сприяє накопиченню нових ситуацій у БЗ і подальшому розширенню когнітивних можливостей системи.

У процесі функціонування ВУС отримує зворотній зв'язок про ефективність обраної дії у вигляді оновленого значення оцінки, що використовується для корекції оцінок образів у БЗ. Таким чином, система поступово навчається на власному досвіді, модифікуючи структуру знань та зміцнюючи ті зв'язки, які приводять до поліпшення результатів, що відповідає принципам підкріплювального навчання.

Блок вибір дії або, надалі, *блок ухвалення рішень* (БПР) реалізує процедуру ухвалення рішення, засновану на аналізі поточної ситуації, цільових функцій, вмісту БЗ, а також оцінки поточного значення оцінки *St*. Фактична інформація про поточну ситуацію представлена безліччю образів, розпізнаних у нинішній момент блоком ФРО, а інформація про якість поточного стану уявлена оцінкою *St*. Безліч розпізнаних образів визначає в БЗ той її розділ, який адекватний поточній ситуації (ті знання, які істинні в поточних умовах). Відповідно до цільової функції, що припускає прагнення ВУС до поліпшення якості стану ОУ, ВУС вибирає по БЗ ту дію, яка має максимальну суму оцінок образів, що викликаються і витісняються. З безлічі вихідних дій, відповідної вибраній дії *Yj*, конкретна вихідна дія вибирається випадковим способом, що відповідає другій цільовій функції, що передбачає прагнення до отримання нових знань.

Блок визначення часу ухвалення рішення визначає глибину проглядання БЗ залежно від поточної оцінки *St*. Чим вище значення *St*, тим більше образів (у порядку убування модуля їх ваги) може врахувати ВУС при ухваленні рішення, тим менше темп ухвалення рішень. При моделюванні цей блок не використовувався і в даній роботі розглядатися не буде.

У ВУС можуть бути засоби для апріорного аналізу наслідків альтернативних вибираних дій на декілька кроків вперед.

Описаний вище алгоритм становить загальну схему управління, яка реалізується вузлом управління знаннями (ВУС) у межах методу адаптивного автоматичного управління (ААУ). Основна особливість цього підходу полягає в тому, що ВУС самостійно накопичує емпіричні знання про динаміку та поведінку об’єкта управління (ОУ) та приймає рішення, ґрунтуючись на вже сформованій базі знань.

Управління в системі ААУ має адаптивний характер, при якому якість керування покращується з часом — у міру накопичення достовірної інформації про поведінку ОУ в різних ситуаціях. Замість простого реагування на вхідні сигнали, ВУС діє проактивно: він постійно шукає нові способи покращення стану об’єкта в умовах, що змінюються. Це відображає наявність позитивного зворотного зв’язку в управлінні, що відрізняє ААУ від традиційних систем, які переважно базуються на негативному зворотному зв’язку (реакція на помилки).

Така поведінка вказує на наявність внутрішньої активності у ВУС — здатності до самостійного цілеспрямованого пошуку оптимальних дій, навіть без прямої зовнішньої стимуляції.

При проектуванні практичних систем управління може бути доцільним комбінування підходів. Зокрема, пропонується розділення простору ознак на дві області:

Область, у якій доступна апріорна інформація про властивості об’єкта управління. У цій області ефективно функціонують традиційні, детерміновані системи управління, що базуються на точних моделях або заздалегідь відомих закономірностях.

Область з недостатньою або відсутньою апріорною інформацією, де поведінка об’єкта недостатньо вивчена, нестабільна або змінна. У такій ситуації доцільно застосовувати адаптивне управління за методом ААУ, яке дозволяє ВУС самостійно навчатися в реальному часі та забезпечувати прийнятний рівень керування за відсутності повної інформації.

Такий гібридний підхід дозволяє ефективно поєднувати переваги класичних систем та інтелектуальних адаптивних методів, підвищуючи надійність і точність функціонування загальної системи управління.

1.3. Основні визначення

У рамках математичної моделі середовища U, кожен вхідний та вихідний сигнал цієї підсистеми розглядається як випадковий вектор. Сукупність таких векторів, які змінюються у часі, утворює випадковий процес. Таким чином, усі зовнішні дії на систему та її реакції можуть бути описані у вигляді часозалежних стохастичних величин.

Окрім цього, доцільно виділити другу категорію процесів, яка охоплює ті процеси, що формуються як виходи внутрішніх елементів блоків ВУС, також параметризовані випадковими величинами. Зокрема, для блоку формування і розпізнавання образів (ФРО), а також інших підсистем, що реалізовані на основі нейронних мереж, такими процесами виступають вихідні сигнали всіх нейронів, які функціонують у системі.

У повній математичній моделі середовища U сукупність процесів, що включає як виходи нейронів ВУС, так і виходи внутрішніх підсистем самого середовища W, визначається як процес середовища U. Цей підхід дозволяє інтегрувати як зовнішні, так і внутрішні сигнальні взаємодії в єдину формалізовану структуру, що забезпечує цілісне моделювання адаптивної системи управління.Надалі ми користуватимемося наступними позначеннями:

T – кінцевий часовий інтервал життя системи;

$t \in T$ - параметр часу;

$t_0 \in T$ - початковий момент часу роботи ВУС;

$\overrightarrow{X}^i(t)$ – вхідний процес, вхідний процес для ФРО, а значить і для ВУС;

$x_i^i(t)$ – i – ая компоненту $\overrightarrow{X}^i(t)$;

$\{\tilde{\tilde{X}}^i(t) \mid t_0 \le t \le t_1\}$ - реалізація вхідного процесу, або вхідний фільм, визначений на інтервалі часу $[t_0, t_1]$;

$\tilde{x}_i^i(t)$ – i – ая компоненту $\tilde{\tilde{X}}^i(t)$;

$\vec{X}^e(t)$ – процес середовища, вихід блоку середовища *W*;

$x_i^e(t)$ – i – ая компоненту $\vec{X}^e(t)$;

$\vec{X}^f(t)$ – процес ФРО, сукупність виходів всіх нейронів блоку ФРО на інтервалі $t \in T$;

$x_i^f(t)$ – i - ая компоненту $\vec{X}^f{}_{(t)}$;

$y_k(t)$ – процес дій, що управляють, на середовище $y_k \in Y$ $y_k \in Y$ із сторони ВУС, де

Y – безліч допустимих дій на середовище із сторони ВУС;

F – безліч образів апарату ФРО.

1.4. Алгебра образів.

Як операції, алгебри, над образами ми використовуватимемо операції тризначної логіки, яка є розширенням звичайної логіки з двома значеннями: істина і брехня, що позначається далі як 1 і 0 відповідно, і має третє значення: невизначеність або ◊. Тут приведені таблиці для операцій тризначної логіки. Перший стовпець містить значення першого аргументу, перший рядок – другого.

Таблиця 1.4.1

∧	1	0	◊
1	1	0	◊
0	0	0	0
◊	◊	◊	◊

Таблиця 1.4.2

$\vee$	1	0	◊
1	1	1	1
0	1	0	◊
◊	◊	◊	◊

Таблиця 1.4.3

$\Rightarrow$	1	0	◊
1	1	0	1
0	1	1	1
◊	◊	◊	◊

Операція заперечення у рамках логіки, що враховує невизначеність, поводиться таким чином: для значення "невизначено" результатом заперечення також є невизначеність. Для всіх інших (визначених) значень операція заперечення збігається з її поведінкою у класичній двозначній логіці.

У подальшому будемо вважати, що результат логічної операції у момент виникнення невизначеності — це невизначеність за означенням. Такий підхід дозволяє підтримувати узгодженість обчислень у випадках, коли повна інформація про логічний стан відсутня або недоступна.

2 МОДЕЛЮВАННЯ СЕРЕДОВИЩА

Для експериментальної перевірки методу автономного адаптивного управління (ААУ) необхідно створити адекватну математичну модель середовища, яка забезпечує достатній рівень достовірності реакцій і поведінки об'єкта управління, щоб імітація відповідала умовам, прийнятним для даної системи управління. Проте перевірка ефективності алгоритмів — не єдина функція моделі середовища.

Однією з важливих задач, які вирішуються за допомогою моделювання, є формування початкової бази знань ВУС. У випадку, коли ВУС починає роботу "з нуля" і не має жодних емпіричних даних, її потрібно попередньо навчити. У деяких випадках проведення навчання безпосередньо на реальному об'єкті є небезпечним або технічно неможливим, оскільки ВУС навчається методом проб і помилок, і помилка на етапі початкового навчання може призвести до катастрофічних наслідків для всієї системи.

З огляду на це, початкове навчання доцільно проводити в умовах лабораторного моделювання, тобто з використанням віртуального середовища. Чим ближчою до реальних умов буде поведінка моделі, тим ефективніше ВУС зможе сформувати початкову базу знань і тим вища ймовірність успішного функціонування системи у реальному середовищі.

Моделювання середовища може реалізовуватись різними способами. Один із підходів — створення фізичного макета об'єкта управління, який функціонує в умовах, максимально наближених до цільового середовища. В такому випадку ВУС поступово навчається до тих пір, поки не буде досягнуто стійкої поведінки в заданому діапазоні зовнішніх факторів, що дає змогу експериментаторам оцінити живучість та адаптивні можливості системи.

Проте створення реального макета на ранніх етапах може бути надмірно витратним, особливо якщо навіть не визначено повний перелік образів, які ВУС повинна розпізнавати. Крім того, в ході розробки може виникнути потреба у кількох

експериментальних прототипах, що значно підвищує складність і вартість проекту.

Реалізація моделі середовища на ЕОМ дозволяє формалізувати процеси, які відбуваються в реальній системі, з необхідною точністю та рівнем деталізації, достатнім для проведення етапу початкового навчання ВУС. Така модель виконує роль віртуального стенду, на якому система може взаємодіяти з умовним середовищем, приймати рішення, оцінювати наслідки своїх дій і поступово формувати адекватну базу знань без ризику пошкодження фізичного об'єкта чи втрати ресурсу.

У рамках цієї моделі повинна бути реалізована система зворотного зв’язку, яка передає ВУС інформацію про наслідки обраних дій, включаючи зміни у стані середовища, значення оцінки якості управління St, а також вплив випадкових або непередбачуваних факторів. Це дозволяє імітувати динамічну та стохастичну природу реального середовища, зокрема вплив шумів, затримок у сигналах або змін у поведінці об'єкта, які неможливо точно передбачити на етапі проєктування.

Такий підхід створює умови для багаторівневого навчання ВУС. На початковому етапі модель середовища може бути спрощеною, з фіксованими параметрами та ідеалізованими умовами. Поступово модель може ускладнюватися, вводячи нові змінні, стохастичні фактори, нелінійні залежності тощо. Це дає змогу покроково готувати ВУС до роботи в реальному середовищі, розширюючи її досвід та підвищуючи якість прийняття рішень.

Зрештою, створення такої моделі є не лише засобом навчання, а й інструментом дослідження — вона дозволяє оцінювати ефективність різних архітектур ВУС, проводити порівняльний аналіз методів ухвалення рішень, тестувати гіпотези про структуру ознак, критеріальні функції, параметри нейронної мережі тощо.

У наступному розділі буде наведено приклад конкретної реалізації моделі середовища, яка використовувалася для навчання прототипу ВУС у рамках даної бакалаврської роботи.

Зважаючи на ці фактори, доцільно на початковому етапі реалізувати модель середовища у вигляді програмної імітації, наприклад, з використанням електронно-обчислювальної машини (ЕОМ). Такий підхід дозволяє гнучко змінювати параметри середовища, здійснювати контрольовані експерименти та поетапно вдосконалювати як саму модель, так і архітектуру ВУС на основі результатів моделювання.Як одну з моделей середовища для досліджень властивостей ААУ ми пропонуємо узяти кінцевий автомат [НО]. НО є широко відомим, добре вивченим, зрозумілим і зручним при моделюванні середовища об'єктом по наступних міркуваннях: 1) різні стани середовища природним чином відображаються в стани НО; 2) переходи з одного стану середовища в інший під дією ВУС і інших об'єктів природним чином відображаються в переходи НО між станами при читанні вхідного слова. Відзначимо, що серед відомих і поширених НО найбільш відповідними для моделі є автомати Мура і недетерміновані автомати Рабина-Скотта або НРС-автомати. Правда, моделі, засновані на перших, потребують додаткового введення стохастичних джерел, а НРС-автомати потребують модифікації, оскільки реальні середовища є недетермінованими об'єктами. Більш того, недетермінованість моделі середовища необхідна для навчання ВУС. Насправді, якби реакція середовища була повністю детермінованою і залежала тільки від дій на неї ВУС, то ВУС, знайшовши перший закон управління, використовувала б тільки його при виборі дій, що управляють, оскільки по критеріях системи управління краще використовувати хоч який-небудь закон управління і отримати відносно гарантований результат, чим продовжувати пошуки методом проб і помилок. Вийшов би замкнутий порочний круг: система впливає на середовище тільки одним способом, середовище детерміновано реагує на цю дію, ВУС бачить тільки одну реакцію (яка може бути не найкращою) і намагається викликати тільки цю реакцію. Уникнути таких «зациклень» можна за допомогою моделювання недетеминированої реакції середовища.

Приведемо визначення автоматів Мура [НО] і введемо модифіковані НРС-автомати.

(Кінцевий) автомат Мура є п'ятірка A = (*Z*, *X*, *Y*, *f*, *h*). Тут *Z* – безліч станів, *X* – безліч входів, *Y* – безліч виходів, *f* – функція переходів, $f : Z \times X \to Z$ і *h* – функция виходів, $h : Z \to Y$ - сюр"єктивне відображення.

Автомат працює за наступним принципом. Якщо НО знаходиться в деякому стані, то вихід автомата визначається функцією виходу. Вихід автомата інтерпретується в даному випадку як реакція середовища, яке, можливо, з деякими перетвореннями в блоці датчиків може бути подана на вхід апарату формування і розпізнавання образів як двійковий вектор. У кожен момент часу автомат читає вхідне слово, яке інтерпретується як сумарна дія із сторони ВУС і інших зовнішніх об'єктів. Безліч входів може бути ширше чим безліч допустимих дій на середовище із сторони ВУС і включати слова або команди, які можуть подаватися із стохастичних джерел, що знаходяться усередині середовища. По прочитаному вхідному слову і функції переходів визначається стан в наступний момент часу.

(Кінцевий) модифікований недетермінований автомат Рабина-Скотта (МНРС) є сімка A = (*Z*, *X*, *T*, *S*, *F*, *h*, *p*). Тут *Z* і *X* – кінцеві множини (станів і входів відповідно; *X* називають також вхідним алфавітом автомата A); $S \in \pi(Z), F \in \pi(Z)$ (безліч початкових і фінальних станів відповідно); $T = (Z \times Y, Z, T')$, де $T' \in \pi(Z \times Y \times Z)$ (інакше кажучи *T* – багатозначне відображення $Z \times X \to Z$ з кінцевою областю визначення); *h* – те ж, що і для автомата Мура; *p* – функція вірогідності переходів, причому

$$\forall z \in Z \sum_{t : t \in T' \wedge z' \in Z \wedge t = (z, x) \to z'} p(t) = 1. \qquad (2.1)$$

Відзначимо, що ми розглядаємо тільки неалфавітні МНРС, тобто НО, у яких немає переходів для порожнього слова $\Lambda : T' \not\subset Z \times (X \cup \Lambda) \times Z$, а, отже, немає і спонтанних переходів. Відмітною особливістю МНРС є неоднозначність переходів або можливість відповідності одній і тій же парі стан - вхідне слово декількох переходів і приписаної кожному переходу вірогідності. Умова (2.1) означає, що сума вірогідності всіх переходів з будь-якого стану є 1.

Відмінність принципу дії МНРС від автомата Мура полягає в тому, що, коли автомат знаходиться в деякому стані і прочитав вхідне слово, то реалізується один з можливих з даного стану і при даному вхідному слові перехід, при цьому вірогідність реалізації переходу визначається функцією *p*.

Приведені дві моделі середовища з двома різними НО не є еквівалентними і задають різні моделі поведінки. Очевидно, що будь-яка модель з автоматом Мура може бути змодельована моделлю з МНРС, причому зворотне твердження для будь-якої моделі невірне. Автомат Мура простіше в реалізації і дослідженнях, а за допомогою МНРС можна побудувати точнішу модель середовища.

3. АПАРАТ ФОРМУВАННЯ І РОЗПІЗНАВАННЯ ОБРАЗІВ.

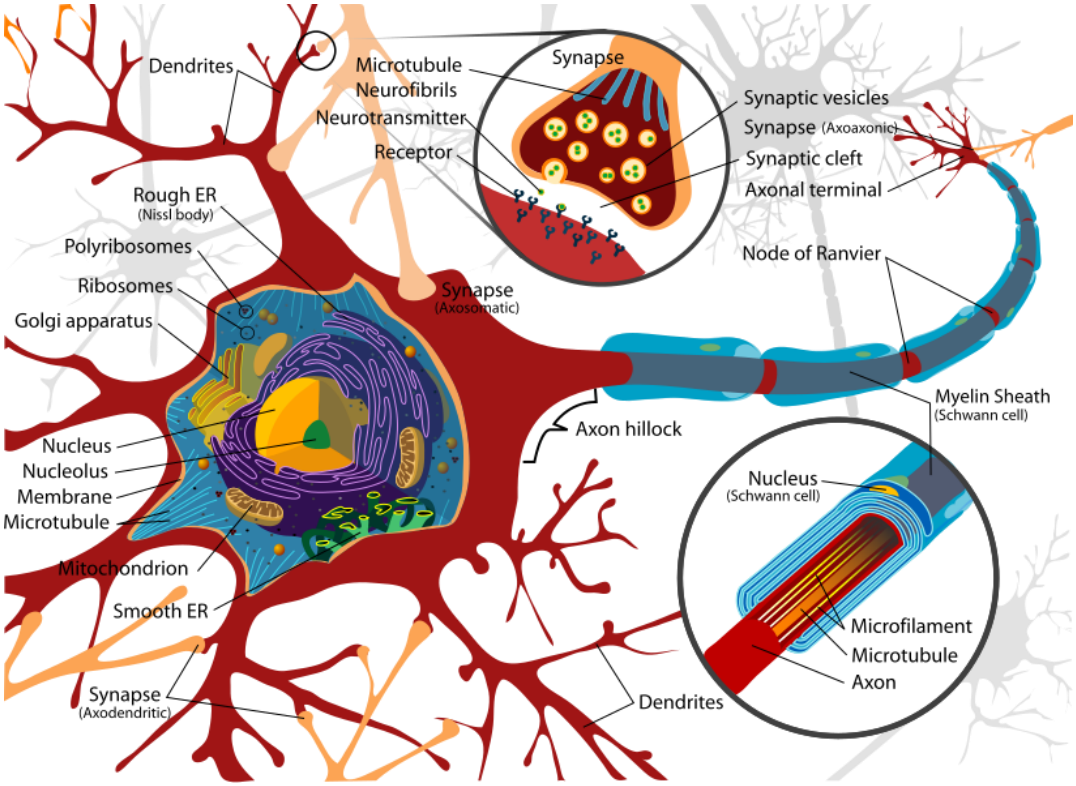


Рис. 3.1. Біологічний нейрон.

На рис. 3.1., представлений в спрощеному вигляді біологічний нейрон. Схемно його можна розділити на три частини: тіло клітки, що містить ядро і клітинну протоплазму; дендріти – деревовидні відростки, що служать входами нейрона; аксон, або нервове волокно, - єдиний вихід нейрона, що є довгим циліндровим відростком, що гілкується на кінці

Для опису **формальної моделі нейрона** виділимо наступні основоположні аспекти:

1. **Набір вхідних сигналів** — кожен нейрон отримує сигнали від інших нейронів або зовнішнього середовища. Вони подаються на вхід у вигляді **вектора** $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, де кожен x_i — числове значення сигналу від відповідного джерела.

2. **Система вагових коефіцієнтів** — кожному входу відповідає вага w_i, яка відображає ступінь впливу відповідного сигналу на активацію нейрона. Ваги утворюють вектор $\mathbf{w} = (w_1, w_2, \ldots, w_n)$.
3. **Активаційна функція** — нейрон обчислює **зважену суму вхідних сигналів**:

$$S = \sum_{i=1}^{n} w_i x_i + b,$$

4. **Вихід нейрона** — результат роботи нейрона, тобто його вихідний сигнал, задається як:

$$y = f(S),$$

де y — це вихід, який може передаватися далі по мережі як вхід для інших нейронів.

5. **Механізм навчання** — параметри моделі (ваги w_i та зміщення b) можуть змінюватися у процесі навчання. Для цього застосовуються **алгоритми оптимізації**, наприклад, метод градієнтного спуску, який дозволяє мінімізувати функцію помилки між очікуваним та фактичним виходом.

Таким чином, формальна модель нейрона визначається як обчислювальний елемент, що виконує трансформацію вхідного сигналу у вихідний за допомогою параметризованої функції. Ця модель є базовим будівельним блоком для побудови більш складних нейронних структур.

3.2. Формальна модель нейрона.

Формальна модель нейрона виникла як спроба математично описати принципи роботи біологічного нейрона, перенісши їх у сферу обчислень і штучного інтелекту. Перші фундаментальні кроки в цьому напрямку були зроблені в середині XX століття.

Однією з найвідоміших перших моделей став персептрон, розроблений Френком Розенблаттом у 1957 році. Ця модель ґрунтувалася на попередній роботі Воррена Мак-Калока і Волтера Піттса (1943), які запропонували першу математичну абстракцію нейрона, що здійснює логічні операції. Мак-Калок і Піттс представили нейрон як простий логічний елемент, що активується при досягненні певного порогу збудження — модель, яка стала основою для подальших розробок.

Модель персептрона Розенблатта значно розширила ці уявлення, дозволяючи навчання нейрона на основі коригування вагових коефіцієнтів відповідно до помилки класифікації. Ця модель отримала велику популярність, хоча й була обмеженою — зокрема, не здатною моделювати логічні функції типу "виключне АБО" (XOR).

Проблема лінійної роздільності персептрона була критично висвітлена у книзі Мінського та Пейперта (1969), що тимчасово зупинило розвиток нейронних мереж. Прорив стався у 1980-х роках з появою багатошарового персептрона (MLP) та алгоритму зворотного поширення помилки (backpropagation), що дозволило ефективно навчати глибші архітектури нейронних мереж.

З подальшим розвитком обчислювальних ресурсів та теоретичної бази, формальні моделі нейронів були розширені — зокрема, з'явилися такі варіанти, як нейрони з локальним зв'язком (у згорткових мережах), рекурентні нейрони, спайкові нейрони, а також моделі, які враховують темпоральну динаміку і нейрофізіологічну достовірність.

На сьогодні формальна модель нейрона є основним будівельним елементом у системах штучного інтелекту, що використовуються в таких галузях, як комп'ютерне бачення, обробка природної мови, робототехніка та адаптивне управління.

Можна вказати цілий ряд основних відмінностей пропонованої в даній роботі моделі і що вже існують. По-перше, в класичних моделях завжди присутній «вчитель» або «супервізор», що підстроює параметри мережі по певному алгоритму, пропонований же нейрон повинен підстроюватися «сам» залежно від «побаченої» ним послідовності вхідних векторів. Формально кажучи, при роботі нейрона повинна використовуватися тільки інформація з його входів. По-друге, в запропонованій моделі немає речовинних вагів і зваженої сумації по цих вагах, що є великим плюсом при створенні нейрочипа і модельних обчисленнях, оскільки цілочисельна арифметика виконується завжди швидше, ніж раціональна і простіше в реалізації. Головна ж відмінність пропонованої моделі полягає в меті застосування.

За допомогою її розв'язуються всі завдання системи, що управляє: формування і розпізнавання образів (ФРО), розпізнавання і запам'ятовування закономірностей (БЗ), аналіз інформації БЗ і вибір дій (БПР), у відмінності від класичних моделей, де розв'язується тільки перше завдання.

Окрім розпізнавання образів, важливою функцією блоку формування і розпізнавання образів (ФРО) в автономних системах є також збереження або запам’ятовування отриманої інформації. Це зумовлено специфікою автономності: у неавтономних системах розпізнані образи можуть зберігатися у зовнішніх носіях або оброблятися поза межами самої системи. Натомість автономна система повинна самостійно забезпечувати збереження розпізнаної інформації.

Загалом, проблема організації пам’яті може бути реалізована різними способами. Одним із відомих підходів є створення кільцевої структури з нейронів, у якій сигнал циркулює або безкінечно, або протягом певного часу до повного згасання. У другому випадку система набуває властивості, аналогічної до "забування", що є характерною рисою біологічних нервових систем. Така здатність до забування є корисною, оскільки дає змогу оптимізувати використання ресурсів, виключаючи з пам’яті інформацію, яка виявилася нерелевантною або застарілою.

У попередніх експериментах використовувалась формальна модель без механізму пам’яті, однак результати показали необхідність її наявності. У зв’язку з цим пропонується впровадження механізму синаптичної пам’яті, який полягає у збереженні сліду вхідного сигналу безпосередньо в синаптичному блоці нейрона. Такий підхід дозволяє створити більш реалістичну модель роботи нейронних структур, а також підвищити ефективність ФРО в умовах обмежених ресурсів та змінного середовища.

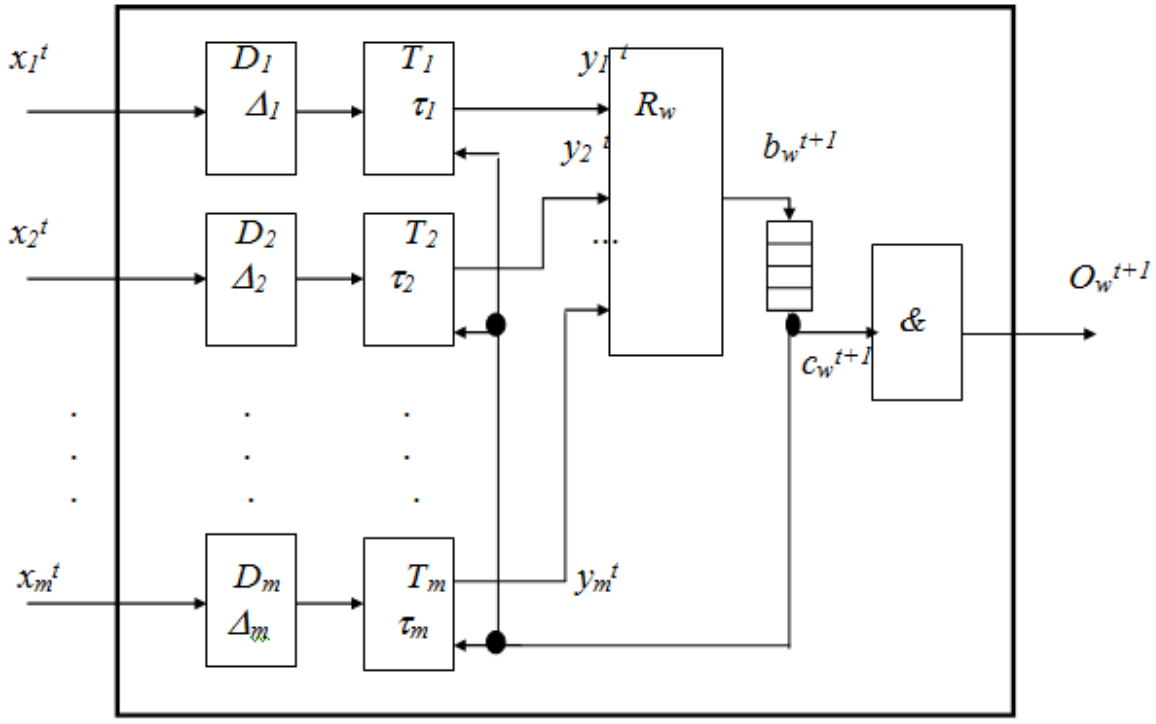


Рис. 3.2. Блокова схема формальної моделі нейрона.

У даній роботі ми використовуємо нейрон , який модифікований . Ми приведемо лише короткий опис. На рис. 3.2. представлена блокова схема пропонованої формальної моделі нейрона. Входи нейрона *xit* подаються на блоки затримки *Di* для затримки сигналу на час $_i$, а потім на тригерні елементи *Ti* для подовження сигналу на величину $_i$. Дані елементи забезпечують деяку невизначеність моменту надходження вхідного сигналу по відношенню до моменту генерації вихідного спайка і утворюють таким чином синаптичну пам'ять, оскільки вхідний сигнал запам'ятовується в цих елементах на деякий час.

З урахуванням затримокΔ Δ_i і $_i$ одержуємо, що, якщо на виході навченого нейрона у момент *t* з'явився одиничний сигнал, то одиничні імпульси на входи нейрона поступили в інтервали часу $di = [t - _i - _i ; t - _i -1]$. Невизначеність моментів надходження вхідних імпульсів буде тим менше, чим менше затримки $_i$.

Приклад тимчасової діаграми роботи навченого нейрона з двома входами і із заданими затримкамиΔ Δ_i і $_i$ ілюструє рис. 3.3. Знаками, питань, показані невизначеності моментів приходу вхідних імпульсів, відповідні інтервалам *di*.

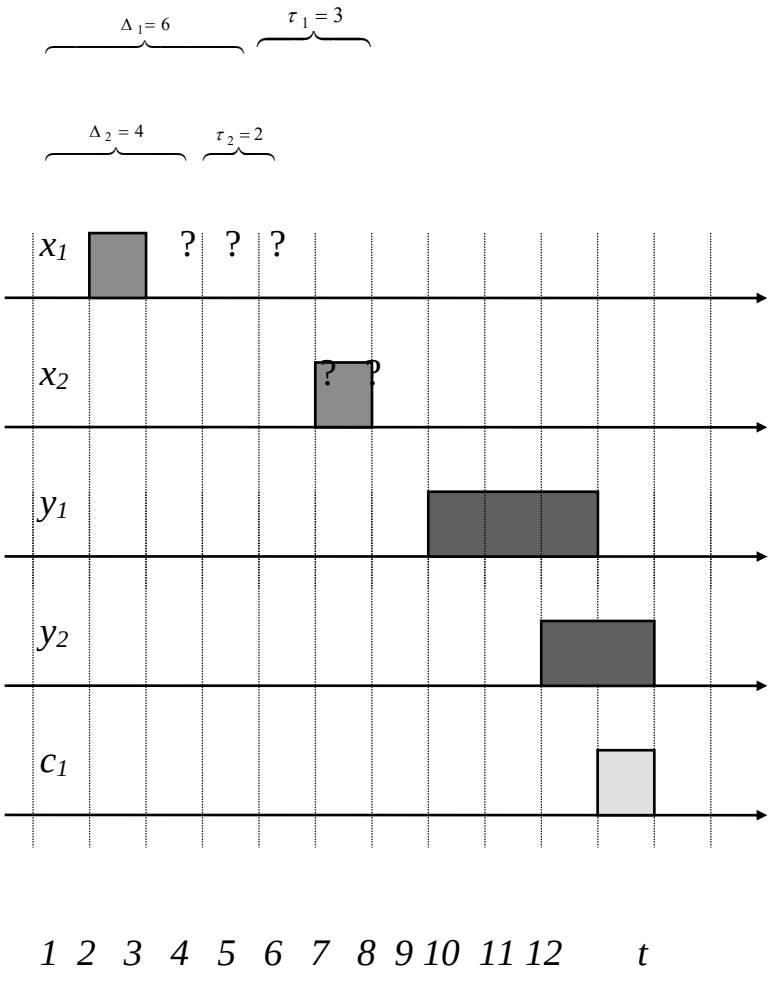


Рис. 3.3. Тимчасова діаграма роботи нейрона.

Різне відношення і розташування затримок $_i$ і $_i$ в часі наділяє нейрон можливостями формування і розпізнавання образів наступних видів.

Якщо $\bigcap_i d_i \neq \varnothing$, то маємо просторовий образ. Наприклад, образ деякої геометричної фігури.

Якщо$\bigcap_i d_i = \varnothing$,, то маємо образ проходження (важливий порядок проходження створюючих, допустима невизначеність в конкретних моментах приходу імпульсів). Прикладом може бути розпізнавання слів при читанні по буквах.

Якщо $\tau \forall \ \tau_i = 1$, то маємо просторово-часовий образ (ППО), в цьому випадку однозначно визначене, по яких входах і в які моменти часу приходили сигнали. Прикладом може бути розпізнавання музичного тону певної висоти.

Елемент *lw* призначений для набору статистики по даному просторово-часовому образу. Значення $l_w=1$ указує на те, що даний нейрон навчений.

Затримки Δ_i і $_i$ задані спочатку, тобто є константними параметрами нейрона. Якщо послідовність сигналів, що поступає на даний нейрон, містить закономірність, що описується такими тимчасовими параметрами, то нейрон сформує образ такого просторово-часового прообразу. Очевидно, що необхідне число нейронів такого типа в мережі буде тим менше, чим більше апріорної інформації про тимчасові характеристики прообразів відомо на стадії синтезу мережі.

Приведемо формалізм нейрона.

$O_w^{t+1} = c_w^{t+1} \ \& \ S_w^{t+1}$;

$c_w^{t+1} = b_w^{t+1} \ \& \ l_w^{t+1}$;

$$b_w^{t+1} = \begin{cases} 1, & if \ \sum_{i=1}^{m} y_i^t / m \geq p_w(N_w^t) \\ 0 \ in \ other \ cases \end{cases}$$

$yit = \overline{/}cwt \ \& ait$;

$$ait = \begin{cases} 1, \ if \ t_{p_i} \in d_i = [t - \Delta_i - \tau_i \ ; t - \Delta_i - 1] \\ 0 \ in \ other \ cases \end{cases};$$

де $t_{p_i} = t \ xit = 1$ - момент імпульсу на вході *xi* ;

$$lw^t = \begin{cases} 1, \ if \ N_w^t \geq M_w \\ 0 \ in \ other \ cases \end{cases};$$

$N_w^0 = 0$;

$Nwt = N_w^{t-1} + bwt$;

Rw(Nwt) - спадаюча сигмоїдальна функція.

3.3. Завдання побудови ФРО.

Для того щоб ефективно та раціонально спроєктувати нейронну мережу блоку формування і розпізнавання образів (ФРО), необхідно зрозуміти зміст або концептуальне значення формованих образів. Іншими словами, важливо визначити, які саме вхідні сигнали сприяють навчанню конкретного нейрона або формуванню конкретного образу.

З цією метою доцільно ввести поняття "повчального вхідного фільму". Під цим терміном розуміється множина вхідних процесів (або послідовностей вхідних сигналів), які при поданні до нейронної мережі приводять до активації певного нейрона або формування цільового образу. Така множина є узагальненою репрезентацією всіх типових ситуацій, які "навчають" даний нейрон розпізнавати конкретну ознаку або патерн.

Виявлення та аналіз повчальних вхідних фільмів дозволяє краще зрозуміти структуру вхідного простору, з яким працює ФРО, а також оптимізувати архітектуру мережі, зменшуючи її розмірність без втрати якості розпізнавання. Це особливо важливо в умовах обмежених ресурсів, а також для забезпечення стабільності та узагальнюючої здатності моделі.

Всякий вхідний фільм $\{\tilde{\tilde{X}}^i(t) \mid t_0 \le t \le t_1\}$ назвемо повчальним для нейрона, якщо t_0 - початковий момент часу роботи системи і $\forall t : t_0 \le t < t_1 \quad \tilde{x}_i^f(t) = 0 \vee \tilde{x}_i^f(t) = \Diamond$ $\tilde{x}_i^f(t_1) = 1$.

Таким чином, завдання побудови НС ФРО можна сформулювати таким чином: для даної сукупності вхідних фільмів побудувати таку мережу, в якій би були присутні нейрони, для яких дані вхідні фільми є повчальними. Побудована таким чином мережа здатна вирішувати еталонну задачу класифікації, де як еталони використовуються дані вхідні фільми. Відома безліч способів конструювання і настройки мереж для класичних формальних моделей нейронів, наприклад, мережі зворотного розповсюдження, що використовують узагальнене δ-правило.

Проблема пропонованого підходу полягає в тому, що 1) система повинна бути автономною, а значить відсутній «вчитель»; 2) взагалі кажучи, a priori не відомі всі життєво необхідні для системи образи. Але якщо ми володіємо апріорною інформацією про умови існування системи (що майже завжди буває), її слід використовувати при конструюванні ФРО.

Можна інакше сформулювати завдання побудови ФРО. Приведемо приклад з системою «Пілот» . У математичній моделі супутника використовуються величини кутового положення супутника φ і його похідної, отже, очевидно, що всілякі поєднання можливих значень цих величин (тобто деяка область на фазовій площині) необхідні для знаходження законів управління системою. Дійсно, допустима система у момент часу *t* знаходиться в змозі $\varphi = \varphi_0, \dot{\varphi} = \dot{\varphi}_0$ і ВУС вибирає деяку дію, що управляє y_k (включення одного з двигунів, наприклад). Ми знаємо, що у момент часу $t + \Delta t$ система опиниться в деякому стані, відповідному крапці на фазовій площині з деякою вірогідністю, де $\vec{x} = (\varphi, \dot{\varphi})$ - крапка на фазовій площині, таким чином, можна говорити про деякий імовірнісний розподіл, що заданий у фазовому просторі і характеризує прогноз поведінки системи через інтервал Δt при виборі дії y_k у момент часу *t*. Якби параметрів $(\varphi, \dot{\varphi})$ було недостатньо для опису законів управління, то функція розподілу залежала б ще і від інших параметрів, і при одних і тих же величинах $\varphi_0, \dot{\varphi}_0$ приймала б інші значення залежно від значень неврахованих параметрів. Отже, ВУС не змогла б не знайти ніякого закону управління, оскільки система шукає статистично достовірну кореляцію між спостережуваним станом ОУ, вибраною дією і станом ОУ через деякий інтервал часу. Законом управління тут ми назвемо сукупність функцій розподілу для кожної дії, що управляє, де Δt знаходиться в деякому діапазоні. Знайдений ВУС закон управління відобразиться в деякому внутрішньому форматі в БЗ, причому він може бути одержаний в процесі навчання системи в реальних умовах прямо під час роботи, або на тестовому стенді, «на землі». Отже, основне завдання побудови блоку формування і розпізнавання образів (ФРО) полягає у створенні образів, які б відповідали необхідному набору параметрів, що характеризують поточний стан системи, а також їх комбінаціям, які мають значення для визначення закону уп-

равління. Інакше кажучи, ФРО має забезпечувати трансформацію вхідних даних у структуровані образи, що є інформативною основою для прийняття рішень у системі управління.

Процес побудови таких образів значною мірою може бути підтриманий використанням математичної моделі об'єкта управління, за умови її наявності. Наявна модель дозволяє визначити суттєві параметри стану, встановити типові сценарії змін, а також сформулювати вимоги до узагальненості й чутливості образів, які формує ФРО. Це, у свою чергу, сприяє підвищенню ефективності навчання нейронної мережі та надійності процесу управління в умовах змінного середовища.

3.4. Розпізнавання просторово-часових образів.

Всяку сукупність значень реалізації вхідного процесу в деякі вибрані інтервали часу $\{\tilde{\tilde{Y}}^i(t) \subset \tilde{\tilde{X}}^i(t) \mid t = t_1, t_2, ..., t_k, t_0 \le t_1 < t_2 < ... < t_k < \infty\}$ називатимемо просторово-часовим чином (ППО).

Один нейрон здатний розпізнавати (тобто здатний навчитися виділяти конкретний ППО серед всіх інших) тільки ті ППО, у яких одиничне значення сигналу для кожної вибраної компоненти вхідного процесу зустрічається не більше одного разу (приклад зображений на верхньому графіку рис. 3.4.). Мережу нейронів можна побудувати так, що в ній формуватимуться будь-які задані ППО (нижній графік рис. 3.4.).

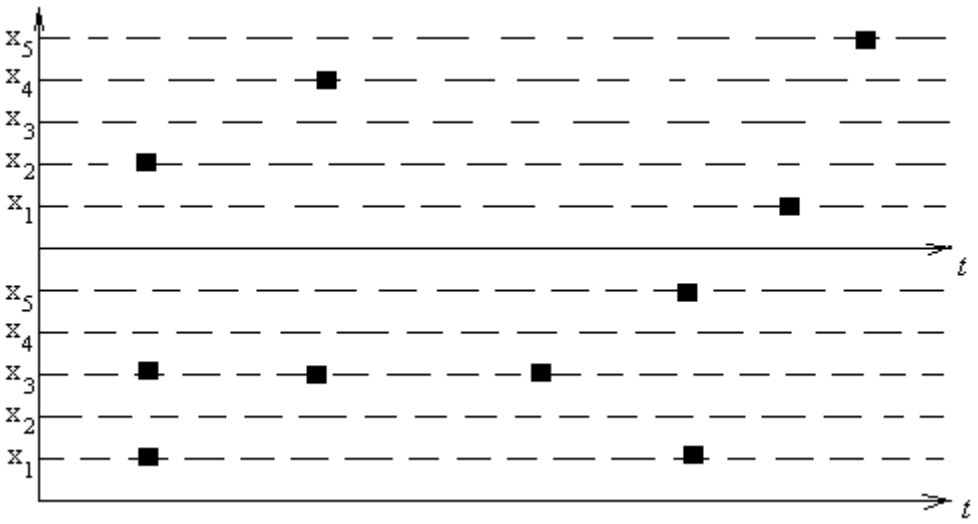


Рис 3.4. Мережа нейронів в якій формуються будь-які задані ППО

4 БАЗА ЗНАНЬ

Процес накопичення знань БЗ в рамках методології ААУ детально розглянутий [4,5]. У цьому розділі буде розглянуто лише основні відмінності запропонованого підходу від аналогічних методик, описаних у попередніх джерелах. Зокрема, розглянемо загальний алгоритм формування бази знань (БЗ) для системи автономного адаптивного управління.

Основною метою даного алгоритму є накопичення статистичної інформації, яка дозволяє встановити причинно-наслідковий зв'язок між діями, обраними системою управління, та реакцією середовища на ці дії. У такий спосіб система поступово формує знання про ефективність конкретних дій у певних ситуаціях, що забезпечує основи для подальшого прийняття рішень.

Другою важливою функцією алгоритму є присвоєння оцінок сформованим образам, що виникають у процесі функціонування блоку ФРО, а також динамічне коригування цих оцінок залежно від значень, що генеруються блоком оцінки стану. Таким чином, база знань постійно оновлюється з урахуванням зворотного зв'язку від середовища, що дозволяє системі адаптувати свою поведінку до умов, що змінюються.

Називатимемо повним від'єднанням ФРО від середовища наступна умова: процеси $\vec{X}^e(t)$ і $\vec{X}^i(t)$ є незалежними. Взагалі кажучи, в діючій системі, звичайно ж ці процеси залежні, наприклад, в простому випадку без блоку датчиків, але для введення деяких понять потрібно в думках «від'єднати» вхідний процес і процес середовища.

Назвемо часом реакції середовища на дію $y_k(t)$ число $I_{y_k} = \min m$, де $m \in Z^+ : \exists i \forall t \in T \Rightarrow$ випадкові величини $x_i^e(t+m)$ і $y_k(t)$ є залежними при повному від'єднанні ФРО від середовища. Закономірністю або реакцією середовища вважатимемо залежність $\vec{X}^e(t)$ від $y_k(t)$.

Іншими словами, час реакції середовища цей час, через який виявляється, тобто може бути розпізнана блоком ФРО, реакція на дію.

Назвемо мінімальною $I_{\min}$ і максимальною $I_{\max}$ інертністю середовища мінімальне і максимальний відповідно час реакції середовища на дію y_k для всіх y_k. Інтервал $[I_{\min}, I_{\max}]$ називатимемо інтервалом чутливості середовища.

Відмітимо, що $0 \le I_{\min} \le I_{\max} \le \infty$.

Введемо сукупність образів

$$b_{ijk} = x_i^f(t-n) \wedge y_k(t-n) \wedge x_j^f(t). \qquad (4.1)$$

Параметр $n > 0$ назвемо запасом на інертність середовища. Сенс b_{ijk} полягає в тому, що якщо b_{ijk} навчений, у нинішній момент часу розпізнаний образ $x^f{}_i$ і ВУС вибере дію y_k те з деякою вірогідністю через n кроків розпізнається образ x_j^f. Аналогічно введемо образ

$$b_{ijk}' = x_i^f(t-n) \wedge y_k(t-n) \wedge \neg x_j^f(t) \qquad (4.2)$$

сенс якого співпадає з сенсом, з тим лише відмінністю, що x_j^f не розпізнається, а витісниться. Оскільки зрештою способом управління ВУС є виклик певних образів і витіснення інших, то сукупність навчених образів $\{b_{ijk}, b_{ijk}'\}$ є матеріалом, сприяючим досягненню мети управління, тобто виклику або витісненню певних образів за допомогою вибору дії y_k з безлічі можливих дій Y на кожному кроці t. Як використовується цей матеріал буде викладено в розділі «Блок ухвалення рішень».

Запас на інертність введений з наступних міркувань. Абсолютно очевидно, що марно намагатися уловити закономірність вигляду «був розпізнаний образ, застосували і через m кроків одержали x_j^f», де, оскільки середовище буде просто не встигати відреагувати. Таким чином, ВУС може уловити закономірності з часом реакції середовища не більшим ніж n. Аналогічно, немає сенсу вибирати n дуже великим, тобто набагато більшим, ніж $I_{\max}$. З іншого боку, використовуючи синаптичні затримки вхідних сигналів в нейроні, ми можемо відловити будь-яку закономірність з часом реакції меншим або рівнішим запасу на інертність. Дійсно, ми можемо побудувати ФРО так, щоб образи і x_i^f формувалися з потрібними затримками, де m – час реакції середовища. Відмітимо, що a priori нам невідомо час ре-

акції середовища m, тому має сенс лише вибрати параметр n *для* всіх образів однаковим і «напевно» більшим ніж ($I_{\max}$ для цього необхідно скористатися апріорною інформацією про середовище).

Тепер зіставимо кожному образу з ФРО деяке число або оцінку. Хай $S(t)$ – вихід блоку оцінки стану, а $S_i(t)$ – оцінка образу, що виходить по наступному алгоритму:

$S_i(0) = 0$, $S_i(t) = f_t(S_i(t_1), S_i(t_2), \ldots, S_i(t_k))$,

де – деяка «усереднююча» функція, - безліч моментів часу, в які образ x_i^f був розпізнаний. У якості f_t звичайно береться просто середнє арифметичне

$$f_t(S_i(t_1), S_i(t_2), \ldots, S_i(t_k)) = \frac{1}{k}\sum_{j=1}^{k} S_i(t_j).$$

Тепер можна визначити, що таке база знань.

Назвемо базою знань сукупність сформованих образів $\{b_{ijk}, b_{ijk}'\}$ і сукупність оцінок $S_i(t)$ для всіх образів ФРО.

Позначимо об'єднання безлічі всіх образів (4.1) і (4.2) через, де *F* – безліч образів ФРО, *Y* – безліч можливих дій. Назвемо В простором образів БЗ.

5 СИСТЕМА ПОБУДОВИ І ДОСЛІДЖЕННЯ НЕЙРОННИХ МЕРЕЖ

5.1. Актуальність системи

У рамках моделювання компонентів вузла управління знаннями (ВУС), побудованих на основі нейронних структур, було виявлено необхідність створення спеціалізованого інструментального програмного середовища, яке б забезпечувало повноцінну підтримку процесу побудови, аналізу та оптимізації нейронних мереж. Такий інструмент має відповідати ряду функціональних та архітектурних вимог, зумовлених специфікою адаптивних систем управління.

Основною метою створення даного інструменту є надання користувачу графічно орієнтованого середовища, що дозволяє:

формувати бібліотеки шаблонних блоків на основі формальних моделей нейронів;

здійснювати побудову мереж з використанням заданих шаблонів;

імітувати роботу мережі з можливістю перегляду проміжних станів, сигналів і активностей;

виконувати збір статистичних даних про функціонування мережі, необхідних для процесу налагодження, діагностики та навчання.

При створенні або виборі відповідного інструментального засобу були враховані наступні ключові критерії:

1. Відкритість архітектури

Інструмент повинен мати відкриту специфікацію інтерфейсів та форматів обробки даних, що дає змогу розширювати функціональність без зміни ядра системи. Це забезпечує мінімальну взаємозалежність між компонентами, що сприяє гнучкості підтримки та масштабування.

2. Гнучкість і розширюваність

Система повинна підтримувати широкий спектр класів нейронних моделей та їх комбінацій, що дає змогу використовувати її для вирішення завдань різної при-

роди — від управління космічними апаратами до прогнозування економічних показників та підтримки прийняття рішень.

3. Багатоплатформеність

Інструмент має бути максимально незалежним від операційної системи, що дозволяє його використання на різноманітних програмно-апаратних платформах.

4. Адаптованість до задач ААУ

Особлива увага приділяється відповідності інструменту вимогам автономних адаптивних систем, включно з простотою використання, орієнтацією на реальні інженерні задачі, та можливістю функціонування в умовах обмежених обчислювальних ресурсів (наприклад, на персональних комп'ютерах).

5. Економічна доцільність

Інструментальне середовище повинно бути економічно ефективним, тобто забезпечувати необхідний рівень функціональності при мінімальних витратах на впровадження, ліцензування та технічну підтримку.

Таким чином, описаний підхід до побудови інструментального середовища дозволяє забезпечити ефективну розробку, аналіз і тестування нейромережевих компонентів ВУС у процесі створення адаптивних систем управління нового покоління.Аналіз наявних або доступних систем САПР і інших систем (наприклад, LabView або систем з класичними НС), тим або що іншим чином задовольняють першим трем критеріям, показав, що всі вони є або ваговитими, або дуже дорогими, або дуже погано пристосовані до моделювання систем ААУ і ОУ з формальною моделлю нейрона, викладеною в розділі «Апарат ФРО» або до роботи з мережами, що складаються з тисяч нейронів. Таким чином, виникла необхідність в інструменті, який би дозволяв перевіряти ідеї ААУ і створювати прототипи ВУС на НС.

5.2. Загальна концепція системи.

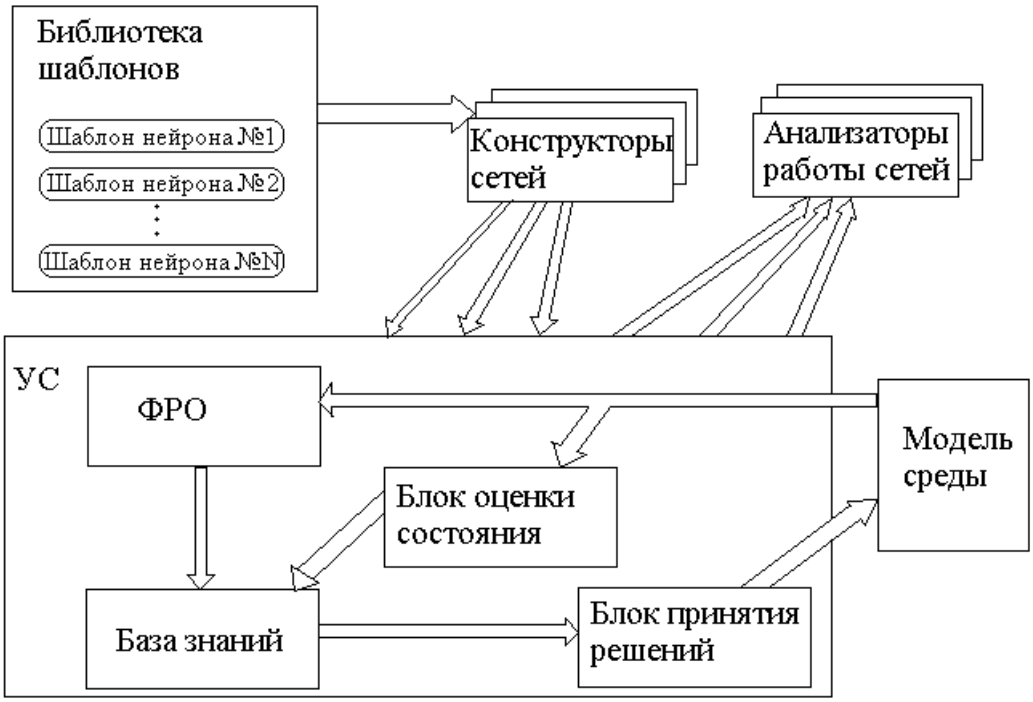


Рис. 5.1. Загальна схема ядра СПДНМ.

На приведеній схемі (рис 5.1) вказані основні класи об'єктів ядра системи і їх взаємодія. Стрілками показані потоки даних при роботі системи. Кожному з основних блоків ВУС відповідає свій блок в системі. Чотири блоки: ФРО, БЗ, БОС і БПР складають ВУС. Нагадаємо, що в підрозділі «Формалізація НС» розділу 1 ми визначили такі поняття як блок, вихідна функція блоку, шаблон, нейронна мережа і формальна модель нейрона. З формальної моделі НС виходить, що блок – це ієрархічна структура, в якій елементи одного рівня сполучені в мережу і кожний з елементів рівня може бути мережею, що складається з елементів нижчого рівня. Розглядаючи вибраний елемент якого-небудь рівня, можна вважати його «чорним ящиком», тобто абстрагуватися від його вмісту і внутрішнього устрою. Наприклад, можна на деякому проміжному етапі конструювання ВУС абстрагуватися від нейро-мережевої реалізації якого-небудь блоку верхнього рівня і спробувати різні реалізації, причому необов'язково нейросетевые. Система не накладає жорстких вимог на внутрішню структуру окремих блоків, що

дозволяє реалізовувати їх як прості функціональні компоненти, без внутрішньої ієрархії чи складної архітектури. На початковому етапі такий блок може бути представлений у вигляді узагальненої функції від вхідних параметрів, яка реалізує відповідну поведінку. У подальшому, у процесі розвитку системи ВУС, внутрішній зміст окремих блоків може бути модифікований або ускладнений — зокрема, набувати ієрархічної структури або реалізовуватись у вигляді нейронних мереж. Принципово важливо, що за умови збереження функціональної еквівентності вихідної поведінки, така заміна не впливає на загальну поведінку системи. Це дозволяє значно спростити проєктування: реалізація системи може здійснюватися поетапно, за принципом «зверху вниз», із використанням тимчасових модулів-заглушок, які в майбутньому замінюються більш складними або спеціалізованими реалізаціями.

Окрім основних функціональних блоків, до складу системи входять ще два класи допоміжних об’єктів:

Конструктори мереж, які забезпечують створення робочих екземплярів нейронних мереж у пам’яті комп’ютера. Вони формують структуру мережі, зокрема на основі специфікації з файлів або інших описових форматів;

Аналізатори роботи мереж, призначені для діагностики, візуалізації та збору статистичних характеристик під час функціонування моделі.

Таким чином, архітектура системи побудована з урахуванням гнучкості, розширюваності та поступової деталізації, що є важливою передумовою для її ефективного розвитку та адаптації до складних прикладних задач.Власне, для кожного джерела і створюється свій об'єкт. (Слід відрізняти дані об'єкти від конструкторів мереж, призначених для створення за допомогою ГІК файли специфікації мереж; ці конструктори в ядро СПДНМ не входять). Специфікація нейронної мережі, яка використовується під час її ініціалізації, може посилатися на шаблони блоків, що зберігаються в бібліотеці стандартних компонентів. Таким чином, бібліотека шаблонів також виступає джерелом для формування структури мережі, забезпечуючи повторне використання перевірених конструкцій та спрощення проєктування.

Важливою складовою інструментального середовища є об’єкти класу “аналізатори”, які мають ключове значення на етапах відлагодження та тестування нейронних мереж. Проблема полягає в тому, що сучасні мережі можуть містити тисячі або навіть десятки тисяч елементів, і кількість тактів їх роботи може сягати сотень і більше одиниць часу. Через це одночасне спостереження за станами всіх елементів є технічно неможливим або надзвичайно неефективним.

Тому виникає необхідність у засобах, які дозволяють агрегувати інформацію про стан мережі, тобто представляти узагальнену характеристику її функціонування у кожен момент часу. Під станом мережі при цьому розуміється сукупність станів усіх її елементів, і ця сукупність має бути інтерпретована у зручній формі для користувача, з можливістю деталізації за вибраними критеріями.

Для вирішення цього завдання використовуються спеціалізовані об’єкти — аналізатори, які здійснюють збереження історії змін станів вибраних елементів протягом заданих інтервалів часу. Надалі ця історія може бути проаналізована з метою виявлення статистичних характеристик, трендів, аномалій тощо.

Кожен аналізатор реалізує свою логіку агрегування і обробки інформації, що дозволяє адаптувати вибір інструмента відповідно до конкретних цілей дослідження або вимог користувача щодо типу й обсягу інформації. Такий підхід забезпечує гнучкість і масштабованість у процесі аналізу функціонування великих нейронних мереж.Відзначимо тут на наш погляд дуже корисну класифікацію об'єктів на інструменти і матеріали . Матеріалами називаються об'єкти, що є свого роду контейнерами інформації і що містять методи тільки для накопичення і нескладних перетворень цієї інформації. Інструментами називаються об'єкти, призначені для обробки матеріалів, тобто для більш інтелектуальних і складних перетворень тієї інформації, яку зберігають об'єкти - матеріали. Таким чином, з погляду цієї класифікації, ми вважаємо нейронні мережі (блоки) матеріалами, а конс-

труктори і аналізатори – інструментами. Слід не плутати ці інструменти-об'єкти з инструментами–приложениями, надбудовами, що є, над ядром.

У реалізації програми ми істотно використовували ідеї об'єктних шаблонів . Далі, в описі реалізації системи ми використовуватимемо російськомовні аналоги термінів, тому, щоб не виникло плутанини, відзначимо, що Фабрика відповідає Factory, об'єктні шаблони – design patterns, Синглетон – Singleton, Chain of Responsibility – Ланцюжок Обробників. Відзначимо, що ідея шаблонів в програмуванні і computer science виявилася вельми плідною і слово «шаблон» тут ми використовуємо в трьох різних сенсах: об'єктний шаблон (design pattern), просто шаблон (у сенсі визначення 1.x.5) і C++ - шаблон (template).

Проходження принципам відвертості припускає закладання можливості розвитку системи через додавання надбудов над ядром (рис. 5.2). Зокрема, одним з напрямів розвитку ми бачимо створення конструкторів бібліотек шаблонів (а, отже, і мереж) за допомогою ГІК. Передбачається, що вихідним продуктом цих конструкторів будуть файли специфікації шаблонів, з якими вже уміє працювати ядро, з яких і формуватимуться бібліотеки шаблонів. Далі, можна було б створити тривимірний візуалізатор БЗ (про це далі), також ми вважаємо, знадобиться окремий інструмент для конструювання самих БЗ, а, можливо, при певному рівні складності блоків ВУС, і для кожного з них по окремому інструменту, які б зважали повною мірою на специфіку блоків ВУС.

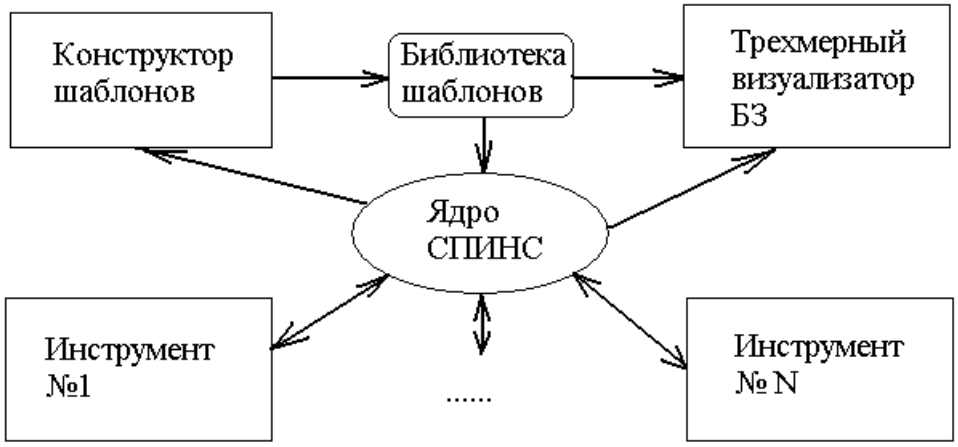


Рис. 5.2.

5.3. Конструктори мереж. Бібліотеки шаблонів.

Конструктори мереж є спеціалізованими програмними об’єктами, призначеними для створення робочих копій нейронних мереж у пам’яті обчислювальної системи. Їх основною функцією є перетворення специфікації мережі, яка може бути задана у вигляді конфігураційного файлу, скрипту або графічної схеми, у відповідну внутрішню структуру, придатну для подальшої імітації, аналізу та навчання.

До ключових функцій конструкторів належать:

Інтерпретація специфікацій мережі, які можуть містити посилання на шаблонні блоки;

Ініціалізація елементів нейронної мережі (нейронів, з’єднань, ваг, функцій активації);

Побудова внутрішніх структур зв’язків між блоками відповідно до логіки заданої архітектури;

Оптимізація структури з урахуванням продуктивності та обмежень на обчислювальні ресурси;

Можливість повторного використання типових конфігурацій за допомогою шаблонів.

У загальному випадку конструктори реалізують архітектурну незалежність, тобто можуть працювати з різними форматами опису (наприклад, JSON, XML, власні формати), і підтримують гнучке масштабування, що дозволяє створювати як прості, так і надзвичайно великі нейронні мережі.

Як вже було відмічено, конструктори мереж ядра СПДНМ призначені для створення внутрішнього представлення мережі в пам'яті комп'ютера за різними джерелами. Тут буде розглянутий тільки один - конструктор по файлу-специфікації мережі, але ми не виключаємо можливості створення конструкторів, що використовують інші джерела.

Бібліотеки шаблонів є структурованими сховищами заздалегідь визначених шаблонів блоків нейронної мережі, які можуть повторно використовуватись під час побудови різних архітектур. Кожен шаблон описує типовий блок, що має фіксовану або параметризовану структуру, набір входів і виходів, типи нейронів, а також алгоритми обробки сигналів.

До прикладів шаблонів можна віднести:

- Одношаровий персептрон;
- Блок згорткової обробки (Convolutional block);
- Рекурентний блок з пам’яттю (LSTM, GRU);
- Блок класифікації або регресії;
- Синаптичні шаблони з пам’яттю або затримками сигналу.

Використання бібліотек шаблонів дає низку переваг:

- Прискорення процесу проєктування за рахунок повторного використання перевірених рішень;
- Зменшення ймовірності помилок при конструюванні мереж;
- Підвищення модульності та уніфікації архітектури;
- Забезпечення узгодженості між різними реалізаціями одного й того самого функціонального блоку.

У процесі моделювання шаблони можуть бути адаптовані або розширені, а також служити основою для створення нових спеціалізованих блоків з урахуванням вимог конкретного застосування. Таким чином, взаємодія конструкторів мереж та бібліотек шаблонів забезпечує структуровану, гнучку та масштабовану основу для створення складних нейронних систем, що особливо важливо в умовах динамічного розвитку систем автономного адаптивного управління.

Конструктор по суті своїй є фабрикою об'єктів класу ЕлементМережі. Ідея фабрики полягає в наступному. Оскільки конструювання мережі полягає в породженні величезного числа різнорідних об'єктів ЕлементМережі, то необхідний об'єкт для регулювання процесу породження і смерті цих об'єктів, або фабрика елементів мережі. Тобто на Фабрику також покладені функції складальника сміття. Регулювання або управління процесом породження полягає в наступному. Ми

маємо багато різних нащадків класу ЕлементаМережі, наприклад, Нейрон, який, в свою чергу має декілька підкласів, соотвествующих кожного з різновидів формальних моделей, а також інші елементи мережі Блок, Джерело, що має також декілька своїх підкласів і т.д. Припустимо, ми модифікували або створили нову версію класу *A* з перерахованих класів - A'. Тоді в кожному місці початкового тексту ми повинні замінити оператора породження *A* на оператора породження A'. Гнучкішою є наступна схема. Фабрика1 уміє, або точніше виражаючись, має методи для породження об'єктів класів A, *B, C* і т.д. При повідомленні про породження, наприклад, об'єкту типу A, вона породжує насправді об'єкт нащадка A: A', а Фабрика2 породжує в даному випадку A''. Таким чином, заміною тільки фабрик ми можемо міняти класи породжуваних об'єктів. Відзначимо, що фабрика на мові C++ природним чином реалізується через C++ - шаблон (*template*) і параметризується типом породжуваних об'єктів. Посилання на ФабрікуЕлементовМережі, що уміє породжувати кожного з кінцевих нащадків ЕлементМережі, зберігає об'єкт Мережа. При ініціалізації КонструктораМережі йому повідомляється посилання на Мережу. Природно, Мережа ще не містить ЕлементівМережі, але вже повинна мати посилання на ФабрікуЕлементівМережі. При конструюванні мережі по файлу специфікації КонструкторМережіПоФайлу (підклас КонструктораМережі) використовує методи породження об'єктів ФабрікиЕлементівМережі, посилання на яку він бере у Мережі.

Відзначимо тут, як вирішена була проблема передачі параметрів конструктору (ініціалізатору, особливому методу, що викликається першим після розміщення об'єкту в пам'яті) елементу мережі і, взагалі, конструктору будь-якого об'єкту, породження і видалення якого знаходиться під управлінням фабрики. Проблема полягає в уніфікації типу передаваних параметрів: вони повинні бути одні і ті ж для всіх типів елементів мережі. Був введений клас Атрибут і методу породження об'єкту у фабрики і, відповідно, конструктору об'єкту передавався список Атрибутів. Кожен Атрибут має ім'я, і конструктор кожного елементу мережі розпізнає тільки деяку підмножину підкласів Атрибуту, «свої атрибути», які дізнається по імені. Наприклад, АтрібутЙмовірності є підкласом Атрибуту, має своє поле раці-

онального типа вірогідність. Конструктор Нейрона2 розпізнає АтрібутЙмовірності в переданому списку атрибутів і використовує значення його поля вірогідність для ініціалізації Нейрона2. Для атрибутів також знадобилася ФабрікаАтрибутів.

Під час розробки Фабрики об’єктів було застосовано ще один об'єктно-орієнтований шаблон проєктування — Singleton (Синглетон). Його основною метою є гарантоване створення лише одного екземпляра певного класу протягом усього часу роботи програми, а також керування централізованим доступом до цього єдиного екземпляра з будь-якої частини системи.

Щодо специфікації нейронної мережі, то під час проєктування мови її опису були поставлені наступні ключові цілі:

1.Забезпечення максимальної гнучкості, тобто здатності описувати широкий спектр можливих архітектур нейронних мереж;

2.Простота синтаксису і легкість сприйняття, що дозволяє користувачеві легко читати, редагувати та розуміти специфікації;

3.Компактність опису, тобто прагнення до мінімізації довжини специфікацій без втрати інформативності;

4.Можливість подальшого розвитку мови, що дозволяє розширювати її конструкції відповідно до зростаючих потреб моделювання.

Як базовий приклад була реалізована нейронна мережа, що складається з чотирьох нейронів, кожен з яких здатен формувати образ, відповідний одному з чотирьох можливих станів навколишнього середовища (НО). Цей приклад демонструє елементарну, але функціональну структуру, яка може бути використана для перевірки працездатності мовних конструкцій та механізмів побудови мережі.Для даного прикладу специфікація має наступний вигляд:

```
[Meta]set for Neuron2 synonym Nset for Brancher synonym I[Inputs]I1,I2,I3,I4[Outputs]I1,I2,I3,I4,N1,N2,N3,N4
[Net Topology]
set for N default connection attribute delay=0
set for N default attribute study_counter=3
DecisionMaker(actions=0,1)
StochasticSource(probability=0.1,value=2)
```

```
Max[DecisionMaker,StochasticSource]
Env[or]
I1[Env(contact_number=0,delay=1)]
I2[Env(contact_number=1,delay=1)]
I3[Env(contact_number=2,delay=1)]
I4[Env(contact_number=3,delay=1)]
N1[I2,I3,I4]
N2[I1,I3,I4](study_counter=4)
N3[I1,I2,I4]
N4[I1,I2,I3]
```

Рис 5.3. Специфікація мережі

Специфікація складається з секцій. Секція починається з вказівки імені секції в квадратних дужках і складається з операторів специфікації. У *Meta* секції зібрані оператори, область застосування яких – вся специфікація, тобто всі секції. Тут, наприклад, можна задати імена–синоніми для шаблонів. У багатьох секціях може з'являтися оператор set. Звичайно, його синтаксис такий:

set for <им»я-приймача> <що-установити> <значення>.

Наприклад, *set for* Neuron2 *synonym* N встановлює ім'я-синонім N для шаблону Neuron2. У секціях *Inputs* і *Outputs* просто перераховуються входи і виходи мережі. Найбільша секція, звичайно, *Net Topology*, де описується топологія мережі. У мові специфікації нейронної мережі посилання на конкретний елемент мережі формується як конкатенація (злиття) двох компонентів:

- ✓імені типу елемента (або його синоніма),
- ✓порядкового номера елемента (нумерація є окремою для кожного типу).

Опис топології мережі реалізується за допомогою спеціальних операторів, у яких зазначається певний елемент мережі та його вхідні з’єднання, що вказуються в квадратних дужках. Кожен вхід, у свою чергу, може бути доповнений списком атрибутів зв’язку, який записується в круглих дужках після позначення входу. Цей список має структуру асоціативного списку типу "ім’я–значення".

Крім того, після переліку всіх входів до елемента, може бути наведено ще один асоціативний список, який містить атрибути самого елемента мережі — та-

кож у круглих дужках. Це дозволяє задати властивості конкретного елемента безпосередньо в його описі.

Кожне з'єднання між елементами може характеризуватися власною множиною атрибутів, які визначають специфіку взаємодії. Прикладом такого атрибута може бути, зокрема, синаптична затримка сигналу.

Варто зазначити, що:

- Кожен тип елемента мережі інтерпретує лише свій набір допустимих атрибутів зв'язку;
- Окрему множину таких атрибутів обробляє ядро системи моделювання;
- Атрибути самого елемента мережі (на відміну від зв'язків) також мають фіксований набір, визначений для кожного типу елемента.

Такий механізм опису забезпечує гнучкість і деталізованість специфікації мережі, що дозволяє повноцінно задавати як структуру, так і поведінкові характеристики моделі.У прикладі специфікації 5.1 використовуються імена деяких вбудованих шаблонів, наприклад *Neuron2* або *Brancher*. Взагалі кажучи, ядром підтримується деяка безліч вбудованих базових шаблонів і в мові є конструкція для визначення нових шаблонів через вже визначені або вбудовані. Інформація про вже певні шаблони зберігається в спеціальному об'єкті БібліотекаШаблонов, що є Синглетоном і що має методи для отримання ідентифікатора шаблону по його імені, витягання інформації про шаблон по його ідентифікатору, а також додавання / видалення шаблонів. Якщо КонструкторСеті при читанні специфікації зустрічає ім'я шаблону або конструкцію визначення нового шаблону, то він звертається до БібліотекеШаблонов для, відповідно, отримання інформації про шаблон по імені або для додавання нового шаблону. Таким чином, користувач може створювати бібліотеки своїх шаблонів, визначення яких зберігаються, наприклад, у файлах і при їх використанні просто включати соотвествующие файли в своїх специфікаціях за допомогою конструкції мови, аналогічної «#include» в мові C / C++. Відзначимо, що бібліотечні файли шаблонів можуть створюватися спеціальним інструментом, конструктором шаблонів.

5.4. Організація обчислень в мережі.

Після створення внутрішнього представлення мережі в пам'яті у вигляді сукупності зв'язаних між собою елементів мережі, мережа готова до обчислень. Обчислення ініціюються з деякої вибраної підмножини елементів мережі, званих виходами мережі. Кожен елемент має свій метод, що реалізовує операцію даної вершини мережі і результат якого інтерпретується як значення вихідного сигналу або значення виходу. Аргументами цього методу є значення виходів у входів елементу в попередні моменти часу, і, можливо, у нинішній момент. При цьому, природно, можливий нескінченний цикл у разі неправильної специфікації мережі. На цей випадок в систему передбачається додати деякий попередній аналізатор коректності топології мережі. Розпаралелювання тут можливе при обчисленні значень виходів для елементів одного порядку, оскільки вони є незалежними.

Оскільки параметр часу в систему введений явно, існує необхідність в повідомленні кожному елементу про настання наступного такту обчислень. При отриманні такого повідомлення, кожен елемент виконує завершувальні операції для даного такту або може просто проігнорувати повідомлення. У реалізації механізму передачі повідомлень використаний об'єктний шаблон Ланцюжок Обробників . Суть його полягає в наступному. Припустимо існує деяка ієрархія класів або ланцюжок, де кожен попередній клас є батьківським для наступного, наприклад ЕлементМережі $\supset$ Нейрон $\supset$ Нейрон2. У ЕлементаМережі визначений (віртуальний в термінах мови C++) метод обробки повідомлення обрабити_Повідомлення*)*. У цьому методі у кожного класу при виклику визначається, чи може даний метод обробити дане повідомлення. Якщо так, то виконується обробка. Потім у будь-якому випадку викликається метод обробки повідомлення батьківського класу, якщо він існує. Наприклад, метод обробки повідомлень у ЕлементаМережі збільшує лічильник тактів (лічильник часу) при отриманні повідомлення НаступнийТакт (нащадок класу Повідомлення). Об'єкт СередовищеЗКінцевимАвтоматом, що є нащадком ЕлементаМережі і КінцевогоАвтомата при отри-

манні даного повідомлення виконує читання вхідного слова, і, природно, викликає обробку повідомлення для своїх батьківських класів.

5.5. Аналізатори роботи мережі.

У процесі моделювання та тестування складних нейронних мереж, особливо у рамках систем автономного адаптивного управління (ААУ), виникає необхідність у засобах спостереження, діагностики та інтерпретації процесів, що відбуваються всередині мережі. Цю функцію виконують аналізатори роботи мережі — спеціалізовані програмні об’єкти, призначені для моніторингу внутрішніх станів і агрегації статистичної інформації про активність елементів мережі.

Основна мета використання аналізаторів полягає у:

- ✓виявленні збоїв або аномалій в роботі окремих елементів або зв’язків;
- ✓підтвердженні правильності функціонування мережі згідно з очікуваною логікою;
- ✓аналізі динаміки навчання;
- ✓оптимізації структури та параметрів моделі;
- ✓зборі даних для подальшого навчання, калібрування або вдосконалення.

У нейромережах великої розмірності (тисячі й десятки тисяч елементів) одночасне спостереження за всіма компонентами є неможливим — як з огляду на обмеження візуального сприйняття, так і з технічної точки зору. Крім того, якщо тривалість роботи моделі охоплює сотні чи тисячі тактів, кількість інформації зростає експоненційно. Тому виникає потреба в інструментах для узагальнення стану мережі у кожен момент часу.

Аналізатори вирішують цю задачу шляхом:

- ✓ вибіркового моніторингу ключових елементів або підмереж;
- ✓ збереження історії зміни станів вибраних компонент;
- ✓ обчислення статистичних показників (середніх значень, піків, тривалостей активності тощо);

✓ формування зведеної інформації з можливістю деталізації за запитом користувача.

Залежно від цілей дослідження, можуть застосовуватись різні типи аналізаторів, зокрема:

1. Аналізатори активності нейронів — фіксують зміни вихідних сигналів нейронів, виявляють періоди збудження або затухання;
2. Аналізатори зв’язків (синапсів) — відслідковують зміну ваг, наявність затримок, блокування або послаблення сигналу;
3. Аналізатори структурної узгодженості — перевіряють топологію мережі на предмет правильності зв’язків, циклічності, ізольованих компонент;
4. Аналізатори навчання — фіксують процес корекції параметрів, динаміку помилки, якість узагальнення;
5. Візуалізаційні модулі — інтегрують інформацію у вигляді графіків, теплових карт, часових діаграм тощо.

Кожен аналізатор є незалежним програмним об’єктом, який може бути активований або налаштований відповідно до поточних потреб користувача. У рамках СПДНМ, наприклад, передбачено можливість підключення зовнішніх аналізаторів, а також розширення їхнього функціоналу через відкриті інтерфейси.

Для відладки мереж часто необхідно знати різноманітну інформацію про стан мережу і окремих її елементів в деякі моменти часу. Мережі передбачаються гетерогенні, тобто що складаються з різних елементів-екземплярів класів-нащадків ЕлементаМережі, і стан кожного елементу в деякі моменти часу може характеризуватися, взагалі кажучи, деяким своїм набором параметрів, крім значення вихідного і вхідного сигналів. Наприклад, НО краще охарактеризувати станом, в якому знаходиться автомат. Тому схема була вибрана наступна: ЕлементМережі є похідним класом від Літописець, який має методи для запису об'єктів Подія в деяку тимчасову послідовність Історія, яку зберігає кожен Літописець. В процесі роботи мережі кожен Літописець записує в Історію один з своїх або загальних об'єктів підкласу Подія. Наприклад, НО записує крім інших подію СтанКА, в якій є поле для вказівки стану НО в даний момент часу. Кожен (нащадок класу) АналізаторРабо-

тиМережі уміє обробляти Історії, витягуючи звідти необхідну інформацію, і потім видаючи її в зручному вигляді на екран. У прикладі 5.7. приведений результат роботи програми, де зображена діаграма вихідних сигналів вибраних ЭлементівМережі, імена яких виведені в першому рядку діаграми. Діаграма одержана як результат обробки Історії вибраних ЭлементівМережі АналізаторомРаботиМережі.

5.6. Реалізація блоку оцінки стану.

Завдання блоку оцінки стану кінець кінцем полягає в зіставленні вихідному вектору середовища *W* деякої оцінки або числа. Фактично, йдеться про завдання деякого функціонала над *BN*, де *N* - розмірність вихідного вектора середовища *W*. Функціонал задається як таблична функція, значення якої перераховуються в спеціальній секції файлу специфікації. Тут указуються типи аргументу і самої табличної функції, може бути вказано значення за умовчанням, а потім перераховуються вхідні і відповідні вихідні значення.

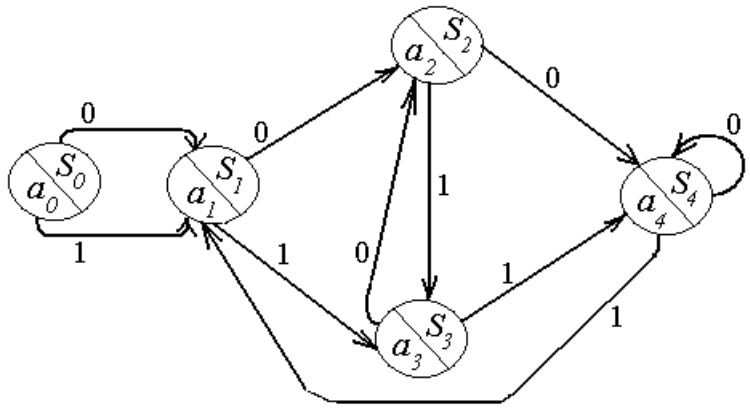
```
[Evaluator]
set type for in BinaryVector
set type for out Integer
set default value for out=0
in=0111,out=15
in=1011,out=10
in=1101,out=5
in=1110,out=20
```

Рис 5.4. Специфікація табличної функції БОС.

Нейромережева реалізація цього блоку нами не створювалася. Передбачається, що її можна побудувати з двох простих підмереж, з яких одна є такою, що розпізнає нейросеть, що формує образи вектора ***in***, а інша - комбінаційну схему з нейронів, що видає сигнал на вихід з номером *out* відповідно до заданого функціонала.

5.7. Реалізація моделі середовища.

Представимо середовище з допомогою НО Мура. Хай НО має чотири стани s1, s2, s3, s4 і представлений діаграмою Мура на рис. 5.5. Файл специфікації для моделі середовища містить опис вихідного сигналу залежно від стану НО (перша частина в прикладі 5.5) і опис самого НО (друга частина). П'ятий додатковий стан НО *initial* є ініціальным. Переходи указуються у вигляді **(**<початковий стан> <вхідне слово>**) ->** <кінцевий стан>.



S_0=*initial*, a_0=0000, a_1=0111, a_2=1011, a_3=1101, a_4=1110

Рис. 5.5

```
[Environment]
initial(output=1111)
s1(output=0111)
s2(output=1011)
s3(output=1101)
s4(output=1110)
[Finite State Automate]
set for word -1 synonym any
(initial,0)->s1
(initial,1)->s1
(s1,0)->s2
(s1,1)->s3
```

(s2,0)->s4

(s2,1)->s3

(s3,0)->s2

(s3,1)->s4

(s4,0)->s4

(s4,1)->s1

Рис 5.6. Специфікація моделі середовища.

5.8. Приклад роботи програми.

Результати роботи програми зручно представити тимчасовою діаграмою, де рядок відображає стан системи у момент часу *ti*. Позначатимемо вертикальним штрихом одиничний сигнал на виході нейрона (N) або датчика (I).

Даний приклад демонструє здатність ВУС знаходити закономірності управління і використовувати їх для поліпшення свого стану. Початковими даними для прикладу були специфікація мережі - рис 5.3, специфікація БОС - рис 5.4, специфікація моделі середовища - рис 5.6.

```
Output signals graph
I1 I2 I3 I4 N2 N3 N4 N1 DecisionMaker Or Evaluator
t =   0  *  *  *  *            1     1    0
t =   1    | | |              1     1    15
t =   2  | |   |              0     0    5
t =   3  |   | |              1     1    10
t =   4  | |   |              1     1    5
t =   5  | | |                1     1    20
t =   6    | | |              1     1    15
t =   7  | |   |              0     1    5
t =   8  | | |                0     1    20
t =   9    | | |              0     1    15
t =  10  | |   |   |          0     0    5
t =  11  |   | |              0     1    10
t =  12  | |   |   |          0     1    5
t =  13  | | |                1     1    20
t =  14    | | |       |      1     1    15
t =  15  | |   |   |          1     1    5
```

t = 16 | | | | 0 1 20

t = 17 | | | | 1 1 15

t = 18 | | | | 1 1 5

t = 19 | | | | 0 0 20

t = 20 | | | | 0 0 20

t = 21 | | | | 0 0 20

t = 22 | | | | 0 1 20

t = 23 | | | | 1 1 15

t = 24 | | | | 1 1 5

t = 25 | | | | 0 0 20

t = 26 | | | | 0 0 20

t = 27 | | | | 0 0 20

t = 28 | | | | 0 1 20

t = 29 | | | | 1 1 15

t = 30 | | | | 1 1 5

t = 31 | | | | 0 0 20

t = 32 | | | | 0 1 20

t = 33 | | | | 1 1 15

t = 34 | | | | 1 1 5

t = 35 | | | | 0 0 20

t = 36 | | | | 0 0 20

t = 37 | | | | 0 0 20

t = 38 | | | | 0 1 20

t = 39 | | | | 1 1 15

t = 40 | | | | 1 1 5

...

Calculation time statistics

Number of net nodes = 11

Time interval length = 600

Calculation time = 1.582 secs

Mean time of calculating one node output = 0.24 ms

Knowledge base statistics

N3 -> N4 with action 1 with probability 141 / 141 = 1

N4 -> N1 with action 0 with probability 141 / 304 = 0.464

N1 -> N3 with action 1 with probability 141 / 141 = 1

Рис 5.7. Результат роботи програми.

На діаграмі виведені вихідні сигнали вхідних елементів (датчиків) I1, I2, I3, I4, нейронів N1, N2, N3, N4, БПР (DecisionMaker), БОС (Evaluator) і внутрішнього елементу середовища (Or), на який подаються сигнали від БПР і стохастичного джерела, а вихід сполучений з входом моделі НО Мура. Безліч можливих дій ВУС на середовище складається з двох елементів, позначених як 0 і 1. З рис 5.6. і 5.4 видно, що станом моделі середовища з найвищою оцінкою є s4. З діаграми 5.7 можна зробити висновок, що ВУС знайшла закономірності управління, достатні для утримання ОУ в змозі s4 (t > 18), але в результаті дії стохастичного джерела після деякого часу перебування в s4 ОУ перескакує з цього стану в s1, звідки ВУС знову переводить його в стан s4.

Роботу системи проілюструємо на рис. 5.8.. У систему входять модель середовища, що складається з НО і Витоку, і ВУС, що складається з блоків ФРО, БОС, БЗ, БПР.

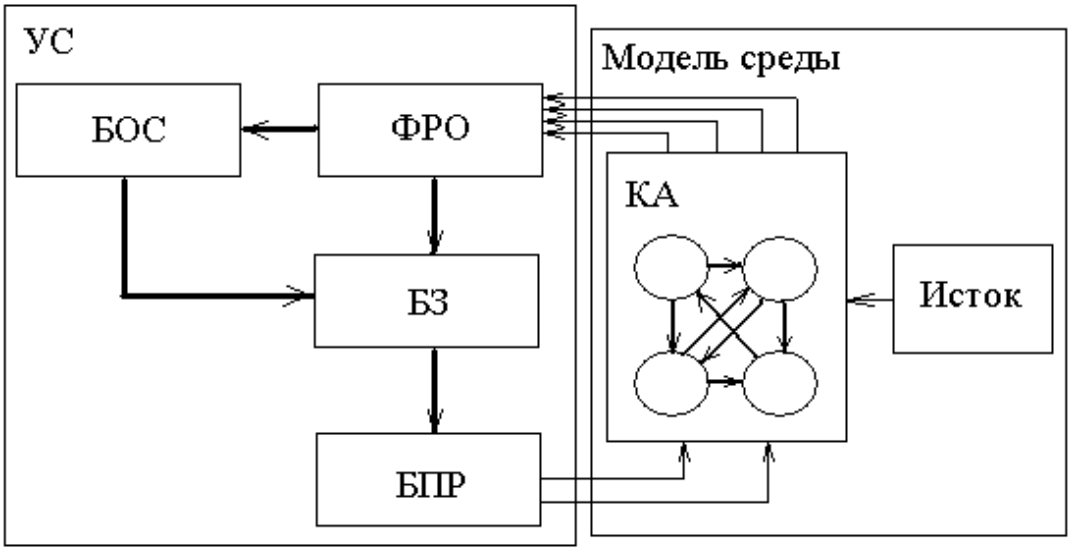


Рис. 5.8. Модель середовища роботи системи

В кінці прикладу виведена інформація про стан БЗ, що містить знання, емпірично знайдені ВУС до моменту закінчення роботи програми.

5.9. Перспективи розвитку СПДНМ.

Окрім намічених в розділі 5 напрямів розвитку системи, а саме створення конструктора мереж з графічним інтерфейсом, розширення мови специфікації мереж і ін., необхідне доопрацювання і розробка нейромережевих реалізацій БЗ і БПР, розробка методів створення реальних додатків по одержаних за допомогою СПДНМ специфікаціях мереж. Привабливим є також створення тривимірного візуалізатора БЗ. Візуалізація БЗ заснована на введенні топології в кінцевому просторі образів бази знань $B = F \times Y \times F$ за допомогою відображення F і Y в R, таким чином, області в B відобразяться в області в R^3.

Якщо образ, $b_{ijk} = x_i^f(t-n) \wedge y_k(t-n) \wedge x_j^f(t)$ сформований, то він відображається точкою кольору, відповідного оцінці, що сформувалася, образу x_j^f. При цьому в просторі позначаться деякі кольорові області (рис. 5.9.), що ілюструють закон управління.

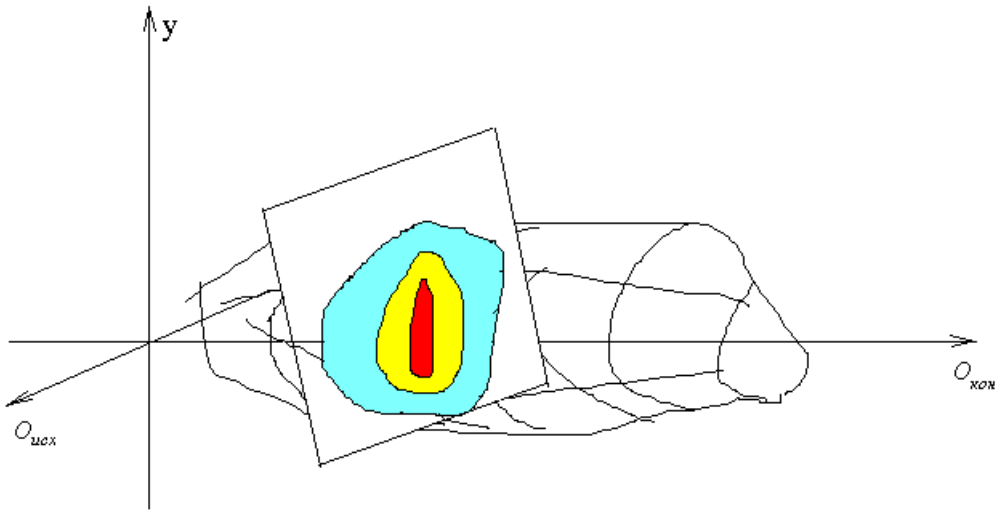


Рис. 5.9. Візуалізація БЗ.

ВИСНОВКИ

У межах бакалаврської роботи були отримані наступні результати:

1. Сформульовано концепцію та реалізовано ядро програмного середовища СПДНМ, призначеного для побудови та дослідження нейромережевих реалізацій прототипів систем управління, що функціонують відповідно до принципів автономного адаптивного управління (ААУ).

2. Проведено удосконалення механізмів формування та розпізнавання образів, алгоритму наповнення бази знань системи управління, а також алгоритму прийняття рішень на основі накопичених знань.

3. Розроблені алгоритми були апробовані в середовищі СПДНМ на низці модельних прикладів, що дозволило здійснити експериментальну перевірку їх працездатності.

4. Отримані результати демонструють, що вузол управління знаннями (ВУС), побудований на основі методу ААУ, здатен самостійно знаходити стратегії управління об’єктом (середовищем) та здійснювати управлінський вплив відповідно до заданих цільових функцій.

СПИСОК ЛІТЕРАТУРИ

1. Холявкіна Т., Троцький Я., Моденов Ю. Адаптивні алгоритми управління технології IoT в умовах обмежених ресурсів

2. Руденко О., Безсонов О., Лебедєв О. Адаптивне управління нелінійними об’єктами з використанням FCMAC

3. Каплун В., Макаревич С., Петренко А. та ін. Адаптивне управління електроспоживанням у мікроенергетичних системах

4. Сінчук О., Бойко С. Нейронні мережі для управління електропостачанням комбінованих мереж

5. Толубко В., Мельник Ю., Макаренко А. Нейромережеве управління нестаціонарним об’єктом

6. Мельник Ю., Кільменінов О., Макаренко А. Модель управління телекомунікаційною мережею на основі НС

7. Каплун В., Макаревич С., Петренко А., Кругляк Г., Кулибаба Є. Адаптивне управління електроспоживанням за умов полігенерації

8. Тимошин Ю., Шевченко М. Система інтелектуального управління для групи роботів .

9. Jamshidi P. et al. Fuzzy self-learning controllers for elasticity management, IEEE Cloud Computing, 2016.