МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ
Національний аерокосмічний університет ім. М.Є. Жуковського
“Харківський авіаційний інститут”

І.В. Шостак, О.С. Топал, О.М. Устінова

ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ КЕРУВАННЯ СКЛАДНИМИ ОБ’ЄКТАМИ

Навчальний посібник

Харків „ХАІ” 2005

УДК 004.78

Інтелектуальні системи керування складними об’єктами / І.В. Шостак, О.С. Топал, О.М. Устінова. – Навч. посібник. – Харків: Нац. аерокосм. ун-т «Харк. авіац. ін-т», 2005. – 72 с.

Розглянуто важливі напрями робіт у галузі створення й експлуатації інтелектуальних систем, а саме знанняорієнтовані моделі й методи керування складними об’єктами, а також методи й алгоритми навчання нейронних мереж. Крім традиційних і широко відомих напрямів робіт у різних галузях систем штучного інтелекту знайшли відображення і дослідження авторів у галузі синтезу інтелектуальних інтегрованих систем керування на основі динамічних експертних систем; викладено принципово нові методи та стратегії виведення на знаннях у таких системах.

Для студентів напряму «Комп'ютерні науки». Може бути корисним і для фахівців, що працюють в галузі систем штучного інтелекту.

Іл. 21. Табл. 12. Бібліогр.: 34 назви

Рецензенти: д-р техн. наук, проф. Н.В. Шаронова,
канд. техн. наук, доц. І.Ю. Шубін

ЗМІСТ

ВСТУП

Основні поняття та визначення. основні напрямки робіт у галузі систем штучного інтелекту

Сучасний етап розвитку систем підтримки прийняття рішень характеризується все більшою інтелектуалізацією процесів прийняття рішень, причому процес автоматизації творчих процесів прийняття рішень йде як ушир, так і вглиб, охоплюючи все нові та нові сфери, які раніше вважалися цілком прерогативою людини.

Зараз існує багато визначень системи штучного інтелекту (ШІ). Дамо одне з них, не претендуючи на вичерпну повноту.

Системою штучного інтелекту називається людино-машинна система, яка використовується для автоматизації процесів творчої діяльності людини в різних сферах, зокрема прийнятті рішень.

Зробимо коротенький огляд історії робіт в галузі систем штучного інтелекту.

Термін штучний інтелект (artificial intelligence) вперше був запропонований у 1956 році на семінарі з аналогічною назвою в Дартмутському коледжі (США) [5]. Семінар було присвячено розробці методів розв'язання логічних задач (в протилежність обчислювальним).

Невдовзі після визнання ШІ окремою галуззю науки відбулося її розділення на два напрямки робіт – нейрокібернетику та кібернетику «чорної шухляди» (black box) [5]. Основну ідею першого напрямку можна сформулювати таким чином: єдиний у світі об'єкт, здатний мислити – це людський мозок, тому будь-який об'єкт (система), який має на меті реалізувати процеси мислення, повинний відтворювати структуру мозку.

Отже ***нейрокібернетика*** орієнтована на апаратно-програмне моделювання структур, подібних до структури людського мозку.

Основою для цього напрямку робіт стали дослідження нейрофізіологів з вивчення структури та функцій мозку. В результаті цих досліджень з'явилися перші наукові роботи стосовно штучних нейронних мереж, які складаються з множини (набору) нейроподібних елементів.

Перші дослідження нейронних мереж були виконані Мак-Калоком (1956 р.) та Фр. Розенблатом (1962 р.). Фр. Розенблат створив нейронну мережу, яка моделювала роботу системи людського зору – «око-зоровий центр мозку» – і отримала назву *„перцептрон”* (від латинського слова «perceptio» – сприйняття). Перцептрон складається з трьох типів нейронів: *S*-елементів (сенсорних), що перетворюють оптичні сигнали на електричні; *A*-елементів (асоціативних), які обробляють сигнали від *S*-елементів і виконують аналіз зорової інформації; *R*-елементів (реагуючих), які виконують класифікацію. Проведені Фр. Розенблатом досліди з перцептроном показали його здатність розпізнавати прості геометричні фігури, а також латинські літери, тобто перцептрон дійсно може служити моделлю зорової системи людини. Ці піонерські дослідження Фр. Розенблата були підхоплені багать-

ма вченими у різних країнах світу і дали початок новому науковому напряму з нейронних мереж. Проте згодом, після наступних досліджень, були виявлені суттєві недоліки тришарового перцептрона, зокрема його нездатність виконувати функції абстрагування й екстраполяції зображень, за якими проводилися навчання, на інші подібні зображення, а також відсутність інваріантності до зсуву та повороту зображень (об'єктів). Теоретичні дослідження математиків М. Мінського та П.В. Новікова показали суттєву обмеженість можливостей перцептронів, зокрема з розпізнавання зображень. У результаті весь напрямок по нейромережам перцептронного типу було піддано нищівній критиці з боку математиків, і роботи в напрямку нейромереж були необґрунтовано припинені на 10 років.

Лише в середині 80-х років вони були знову відновлені, і зараз цей напрямок переживає новий бум.

У 1980-х роках в Японії в рамках проекту ЕОМ 5-го покоління було створено перший нейрокомп'ютер. На цей час обмеження за ємністю пам'яті та швидкодією комп'ютерів були практично зняті. З'явилися так звані трансп'ютери – мікроЕОМ з великою кількістю паралельних процесорів. Вони почали широко використовуватися в нейромережах.

Можна виділити три підходи до створення нейронних мереж: 1) апаратний; 2) програмний; 3) гібридний.

Апаратний підхід базується на розробленні інтегральних схем, в яких відтворюється структура нейронної мережі. В останні роки з'явилось багато фірм, що розробляють та виготовляють нейрочіпи, в яких реалізовано паралельну архітектуру нейромережі.

Програмний підхід пов'язаний з розробкою програмних моделей нейромереж на звичайних ЕОМ. Основною областю застосування нейрокомп'ютерів стали системи розпізнавання образів.

Кібернетика «чорної шухляди». В основу цього підходу покладено такий принцип: «Не суттєво, як влаштована інтелектуальна система. Головне, щоб на задані вхідні стимули (сигнали) вона реагувала так само, як і людський мозок». Цей напрямок спочатку був орієнтований на пошук алгоритмів розв’язку інтелектуальних задач на існуючих на той час комп'ютерах.

Суттєвий внесок в становлення нової науки (напрямку) внесли такі вчені, як МакКарті, автор першої мови програмування для задач штучного інтелекту – ЛІСП, та М. Мінський, автор ідеї та винахідник фрейму та фреймової моделі подання знань.

У середині 60-х і 70-х років проводились інтенсивні пошуки моделей і алгоритмів, здатних вирішувати інтелектуальні творчі задачі. У 60-ті роки народилась ідея лабіринтного пошуку, з'явились перші програми для гри в шахи і шашки. 60-70-ті роки – це епоха розвитку евристичного програмування. Родоначальниками цього цікавого напрямку були вчені Ньюелл, Саймон та Шоу, які розробили систему GPS (General Problem Solver) – систему загального розв’язання задач. Ця сис-

тема була з успіхом застосована для пошуку доведень теорем з евклідової геометрії на основі системи аксіом.

Програма GPS при пошуку доведень теорем використовує так звані евристики – прийоми для скорочення числа варіантів перегляду, якими користується людина. Програма GPS виявилась дуже ефективною і знайшла доведення не тільки всіх теорем планіметрії, а майже усіх теорем з розділу математичної логіки – обчислення висловлювань.

Евристичне програмування та ідеї програми GPS були з успіхом використані і в інших розділах математики, зокрема у розділі математичної логіки, для автоматичного пошуку доведень теорем у 70-ті роки. У цей же час Робінсон розробив *метод резолюцій,* який дозволяє автоматично доводити теореми з наявного набору аксіом. На основі методу резолюцій Альбер Кальмерое у 1973 р. створив мову логічного програмування ПРОЛОГ. Одночасно радянський математик Ю.С. Маслов запропонував метод зворотного висновку, який дозволяє знайти розв'язок логічних задач, рухаючись від кінця (наслідку) до початку (посилання).

Приблизно в цей же час суттєвий прорив у галузі ШІ було зроблено у США, де в середині 70-х років замість пошуку універсальних алгоритмів мислення виникла ідея моделювати конкретні знання фахівців-експертів. У США з'явились перші експертні системи (ЕС), що базуються на знаннях, були створені ЕС MYCIN (1976) та DENDRAL (1978), у результаті яких виник і почав застосовуватися новий підхід до вирішення задач ШІ, заснований на поданні знань.

Наприкінці 70-х років у сферу штучного інтелекту активно вторгається Японія, яка об'явила про початок проекту ЕОМ 5-го покоління. Цей проект був розрахований на 10 років і об'єднав зусилля кращих молодих спеціалістів провідних компаній Японії в галузі комп'ютерної техніки. У результаті було створено перший нейрокомп'ютер і перші інтелектуальні роботи.

Історія досліджень в галузі штучного інтелекту в Радянському Союзі

1954 року в СРСР в МДУ ім. М.В. Ломоносова почав працювати перший семінар «Автомати і мислення» під керівництвом академіка А.А. Ляпунова, проводились численні роботи в царині розпізнавання зображень, мови.

1974 року створено Наукову Раду з проблеми штучного інтелекту при Президії АН СРСР, яку очолив академік Г.С. Поспєлов. З ініціативи Ради було започатковано п'ять комплексних наукових проектів під керівництвом провідних вчених у сфері ШІ, а саме:

ДІАЛОГ – роботи стосовно розуміння натуральної мови – керівник А.П. Єршов;

СИТУАЦІЯ – ситуаційне керування – керівник Д.О. Поспєлов;

БАНК – створення банків даних і знань – керівник Л.Т. Кузін;

КОНСТРУКТОР – автоматизація конструкторського проектування – керівник О.І. Половінкін;

ІНТЕЛЕКТ РОБОТА – проектування інтелектуальних роботів – керівник Д. Охоцімський.

У 80-ті роки проводяться активні дослідження в галузі експертних систем. 1988 року створено асоціацію штучного інтелекту в СРСР, до складу якої увійшли понад 300 вчених. Президентом асоціації обрали професора Д.О. Поспєлова, відзначивши його винятковий вклад у розвиток багатьох напрямків у галузі ШІ.

В Україні наукові дослідження у сфері штучного інтелекту сконцентрувалися в Інституті кібернетики НАНУ. Тут у 70-80-ті роки проводили роботи стосовно створення нових методів, алгоритмів розпізнавання зображень професори В.А. Ковалевський, М.І. Шлезінгер, В.І. Рибак та інші. У сфері розпізнавання мовних сигналів значний внесок зробив професор Т.К. Вінцюк. Результати їх робіт відомі далеко за межами України. Значний внесок у розвиток ряду перспективних напрямків досліджень у сфері ШІ зробив академік НАНУ О.Г. Івахненко та його учні, які ще в 70-ті роки розробили нові алгоритми навчання та самонавчання розпізнавальних систем, що знайшли відображення в численних монографіях.

Особливо слід відзначити вклад О.Г. Івахненка в створення принципово нового методу індуктивного моделювання складних систем – МГУА, який вже понад 30 років з успіхом використовується не тільки в Україні, але й у провідних зарубіжних країнах (США, Японія, Німеччина) для розроблення моделей складних інтелектуальних систем.

У галузі моделювання мислення та психіки цікаві дослідження були виконані під керівництвом академіка М.М. Амосова, під час яких було розроблено одну з перших моделей психіки і мислення з урахуванням емоціональної сфери. Слід відзначити чималий внесок у галузь нейронних мереж та нейрокомп'ютерів вчених Кібернетичного центру НАНУ, зокрема Е.М. Куссуля, Б. Резніка та інших. В області розроблення інтелектуальних систем планування дій протягом багатьох років плідно працює В.П. Гладун.

В останні три роки вченими Кібернетичного центру започатковано нову комплексну програму НАН України «Образний комп'ютер», основні цілі якої – розроблення та впровадження комп'ютерів, заснованих на принципах ШІ, які будуть здатні оперувати не тільки даними у числовій формі, а й образами – візуальними і мовними. Керівником програми є професор Т.К. Вінцюк.

Як видно з цього огляду історії розвитку штучного інтелекту, на даний час це є перспективною галуззю науки, що швидко розвивається та має багато напрямків.

Виділимо деякі основні з цих напрямків.

1. Подання знань, маніпуляція знаннями та створення експертних систем.

2. Спілкування, комунікації «людина–машина».
3. Розпізнавання образів.
4. Навчання й самонавчання.
5. Планування дій, пошук розв'язків задач.
6. Нейронні мережі.
7. Самоорганізація, методи евристичної самоорганізації.
8. Генетичні алгоритми й еволюційне моделювання.
9. Автоматизація конструювання та проектування нових виробів, пошук винаходів за допомогою ЕОМ.

Дамо стислу характеристику деяких з цих напрямків.

1. Подання знань і маніпулювання знаннями. У рамках цього напрямку вирішуються задачі, пов'язані з формалізацією й поданням знань у пам'яті інтелектуальної системи (ІС). Для цього розробляються спеціальні моделі подання знань та мови для опису знань. Вивчаються джерела, з яких ІС може видобувати знання, і розробляються процедури, за допомогою яких можливе отримання нових знань. Проблема подання знань для ІС дуже актуальна, оскільки ІС – це системи, що спираються на знання про відповідну предметну сферу.

Для того щоб користуватися знаннями, необхідно навчити ІС маніпулювати ними. В рамках цього напрямку будуються засоби поповнення знань, вивчаються системи класифікації знань, розробляються процедури узагальнення знань і формування на їх основі понять. Створюються методи правдоподібного виводу нових знань на основі наявних з використанням різних правил висновку. Даний напрямок глибоко пов'язаний зі створенням експертних систем.

2. Спілкування. До кола задач цього напрямку належить проблема розуміння зв'язних текстів на обмеженій та необмеженій природній мові, розуміння мови, синтез мови, теорія комунікацій «людина–машина». До цього ж кола проблем належать задачі формування пояснень дій інтелектуальної системи, які вона повинна дати на запит людини. На базі досліджень у цій сфері створюються методи побудови лінгвістичних процесорів, систем діалогового типу.

3. Сприйняття та розпізнавання образів. Розпізнавання образів, класифікація – одна з найважливіших властивостей інтелекту як природного, так і штучного.

Термін „розпізнавання образів” рівною мірою стосується як процесів сприйняття та пізнання, так і класифікації об'єктів, що спостерігаються.

Метою створення автоматизованих систем розпізнавання образів є автоматизація процесів сприйняття й пізнання об'єктів, що пов'язано з пошуком, ідентифікацією, класифікацією та описом образів на основі аналізу реальних даних. У задачі розпізнавання образів можна виділити два етапи: 1) аналіз вхідних об'єктів, їх опис у вихідному просторі ознак, пошук інформативних ознак, достатніх для правильного розпізнавання, та перехід до опису об'єктів у просторі інформативних ознак; 2) класифікація об'єкта, тобто віднесення його до одного з кла-

сів. Для цього розробляються відповідні вирішальні правила класифікації. Слід зазначити, що вхідна інформація про вхідні об'єкти може мати самий різний характер: зображення (плоскі та об'ємні), мовні сигнали, електричні, оптичні та інші види сигналів, числові послідовності, деякі сценарії тощо.

4. Навчання. Одна з суттєвих особливостей людського інтелекту – це здатність до навчання. Під навчанням розуміється накопичення досвіду у розв’язанні задач і перенесення його на інші задачі, які досі не вирішувалися.

Для того щоб це стало можливим, необхідно створити методи формування умов задачі за описом проблемної ситуації, або в результаті спостережень за нею, навчитися переходити від відомих рішень (розв'язків) часткових задач (прикладів) до розв’язання загальної задачі; створити прийоми (засоби) декомпозиції початкової задачі на більш малі задачі так, щоб вони виявилися відомими для системи. На цей час існує два основних напрямки у сфері навчання ІС (два типи моделей навчання): перший базується на асоціативній моделі навчання, згідно з якою будь-яке навчання є встановленням асоціативних зв'язків у нейроноподібних структурах; другий – на лабіринтній моделі навчання, згідно з якою навчання – це процес пошуку напрямку руху в лабіринті можливих варіантів (напрямків руху) з оцінюванням перспективності руху в даному напрямку за деякими локальними критеріями оцінки.

Важливий клас задач становлять так звані задачі самонавчання, або навчання без вчителя (non-supervised learning). До цього класу задач належать задачі кластеризації або кластер-аналізу (cluster-analysis). Ці задачі формулюються так: є якась вибірка (множина) об'єктів, класифікація яких невідома; треба розбити цю множину на деякі підмножини (кластери) так, щоб максимізувати (або мінімізувати) вибраний критерій кластеризації. При цьому використовується вибрана метрика відстані між об'єктами, які належать до одного кластера.

5. Планування дій розв’язання задач. Функціонування багатьох ІС носить цілеспрямований характер (наприклад, автономні інтелектуальні роботи). Типовим прикладом такого функціонування є розв’язання задач планування досягнення поставленої мети з деякої фіксованої вихідної ситуації. Результатом розв’язання задачі повинен бути план дії – частково упорядкована послідовність дій.

Такий план нагадує деякий сценарій (граф), у якому як відношення між вершинами виступають відношення типів «ціль – підціль», «ціль – дія», «дія – результат».

Довільний шлях у цьому сценарії, який веде від вершини, що відповідає початковій ситуації, в довільну цільову вершину, визначає план дій.

Пошук плану дій виникає в ІС лише тоді, коли вона зустрічається з нестандартною ситуацією, для якої апріорі немає відомого заздале-

гідь набору дій. Усі задачі побудови плану дій можна розділити на два класи, яким відповідають різні моделі: планування в просторі станів (***SS***-проблема) та планування в просторі задач (***PR***-проблема).

У першому випадку (***SS***-проблема) вважається заданим деякий простір ситуацій. Опис ситуацій містить стани зовнішнього світу та стани ІС. Ситуації утворюють деякі узагальнені стани, а дії ІС призводять до зміни актуалізованих у даний момент станів. Серед узагальнених станів виділяють початкові стани (зазвичай один) та кінцеві (цільові). ***SS***-проблема полягає в пошуку шляху, який веде із початкового стану до одного із кінцевих.

При плануванні в просторі задач (***PR***-проблема) ситуація є іншою. Простір утворюється в результаті введення на множині задач відношень типів «частина – ціле», «задача – підзадача», «загальний випадок – окремий випадок». Проблема планування полягає в пошуку такої декомпозиції вихідної задачі на підзадачі, яка б приводила до підзадач, розв'язок яких є відомим. Цей метод (підхід) приводить до хороших результатів тому, що часто рішення задачі має ієрархічну структуру. Пошук планування в просторі задач полягає в послідовному зведенні початкової задачі до все більш простих доти, доки не будуть отримані лише елементарні задачі. Частково впорядкована сукупність таких задач і складає рішення початкової задачі. Декомпозицію задачі на альтернативні підмножини підзадач зручно представляти у вигляді ***І/АБО***-графа. Будь-яка вершина такого графа, крім кінцевої, має або кон'юнктивно зв'язані з нею дочірні вершини (***І***-вершини), або диз'юнктивно зв'язані (***АБО***-вершини). В окремому випадку при відсутності ***І***-вершини маємо граф простору станів. Кінцеві вершини є або заключними (їм відповідають елементарні задачі), або тупиковими. Початкова вершина (корінь ***І/АБО***-графа) являє собою вихідну задачу. Мета пошуку на ***І/АБО***-графі – показати, що початкова вершина розв'язувана. Розв'язуваними є ті заключні ***І-***вершини, у яких розв'язувані усі дочірні, або ***АБО***-вершини, у яких розв'язувана хоча б одна дочірня. Розв'язуваний граф складається з розв'язуваних вершин і вказує спосіб розв'язання (розв'язуваності) для початкової вершини. Наявність тупикових вершин приводить до нерозв'язуваних вершин. Нерозв'язуваними вважаються тупикові ***І***-вершини, у яких нерозв'язувана хоча б одна дочірня, та ***АБО***-вершини, у яких нерозв'язувані усі дочірні.

Існує багато алгоритмів у просторі задач. Розглянемо, наприклад, метод ключових операторів. Нехай задано $\langle A, B \rangle$, де ***A*** – вихідна задача, ***B*** – кінцева задача і відомо, що оператор ***f*** обов'язково входить до її рішення. Такий оператор називається ключовим. Нехай для застосування ***f*** потрібно мати стан ***C***, а результат його застосування є ***f(C)***. Тоді ***І***-вершина породжує три дочірні: $\langle A, C \rangle$; $\langle C, f(C) \rangle$ і $\langle f(C), B \rangle$, з

яких середня є елементарною задачею. Далі до задач $\langle A, C \rangle$ і $\langle f(C), B \rangle$ також шукають ключові оператори, і вказаний процес редукції повторюється доти, доки це можливо. У результаті вихідна задача розбивається на впорядковану сукупність підзадач (елементарних), для кожної з яких розв'язок відомий.

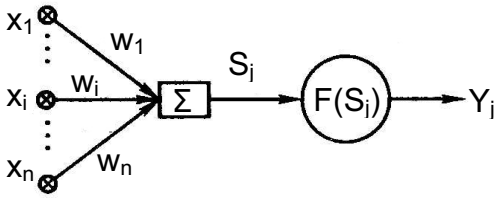


Рис. В.1. Структура нейрона

6. Нейронні мережі – один з найбільш популярних напрямків робіт в області інтелектуальних систем, який веде свій відлік часу з появи перцептрона Фр. Розенблата. Нейронні мережі складаються з великого числа нейроподібних елементів – формальних нейронів.

Кожний нейрон має декілька вхідних зв'язків-синапсів і один вихідний зв'язок-аксон (рис. В.1). Кожний зв'язок ***(i, j)*** характеризується своєю вагою w_{ij}. Якщо на ***i***-му вході подано сигнал x_i, то через синапс ***(i, j)*** на вхід ***j***-го нейрона поступає сигнал $x_i w_{ij}$. Сигнали на вході ***j***-го нейрона сумуються, і сумарний сигнал S_j дорівнює $\sum_{i \to j} x_i w_{ij}$.

Вихідний сигнал ***j***-го нейрона y_j визначається як $y_j = f(S_i)$, де f – так звана функція активації спрацьовування нейрона. Найчастіше використовуються функції активації таких типів:

а) сигмоїд

$$f(S_i) = \sigma(S_i) = \frac{1}{1 + e^{-S_j}};$$

б) релейна

$$f(S_i) = \begin{cases} 1, & \text{якщо } S_j \geq T_j; \\ 0 & \text{у протилежному випадку,} \end{cases}$$

де T – поріг спрацьовування нейрона;

в) пропорційно-релейна

$$f(S_i) = \begin{cases} kS_j, \text{ якщо } |S_j| < T, \ k = \frac{1}{T}; \\ 1, \text{ якщо } S_j \geq T_j; \\ -1, \text{ якщо } S_j \leq -T. \end{cases}$$

Нейронні мережі (НМ) складаються із довільного числа шарів із нейронів.

1. СИСТЕМИ КЕРУВАННЯ СКЛАДНИМИ ОБ'ЄКТАМИ

1.1. Складні об'єкти керування та їх особливості

Керування сучасними технологічними об'єктами, такими, як складні виробничі комплекси, системи енергопостачання, атомні електростанції, пов'язане з необхідністю накопичення й обробки в реальному часі великих обсягів інформації про досвід експлуатації та хід технологічного процесу, поточний стан об'єкта та довкілля. Ця проблема може бути вирішена шляхом створення відповідних інтелектуальних комп'ютеризованих систем, які в своєму складі мають базу знань, бази даних та засоби виведення на знаннях з метою прийняття рішень. Оскільки процеси сучасного виробництва характеризуються високою динамікою та значним зростанням потоків інформації, яку треба обробляти для прийняття рішень. Тут доцільно застосовувати динамічні експертні системи (ДЕС) [1].

ДЕС можуть функціонувати в контурі зворотнього зв'язку з об'єктом і дозволяють враховувати поточний стан технологічного процесу з використанням значних обсягів необхідної інформації. Крім того, ДЕС дають змогу накопичувати та узагальнювати знання і досвід експертів і таким чином підвищувати ефективність керування складним об'єктом.

Науково-технічний прогрес у промисловості, економіці й суспільстві в другій половині XX сторіччя відзначився появою нового класу об'єктів керування – складних об'єктів.

Характерні риси складних об'єктів:

– наявність порівняно великої кількості підсистем, які пов'язані між собою матеріальними, енергетичними, інформаційними потоками;

– неповнота апріорної інформації про стан деяких або загалом усіх підсистем об'єкта, тобто в процесі роботи залежності змінних на вході від змінних на виході суттєво змінюються під впливом збурень з боку довкілля;

– наявність численних управляючих та збурюючих впливів, які в процесі керування мають задовольняти значній кількості обмежень;

– загальна мета визначається низкою критеріїв підсистем (часом суперечливих) та може бути досягнута тільки в результаті взаємодії усіх підсистем складного об'єкта;

– функціонування складного об'єкта може супроводжуватись його розвитком, тобто зміною складу і структури підсистем складного об'єкта.

Ці риси значно ускладнюють задачі керування такими об'єктами, і тому це може здійснюватись, як правило, з використанням обчислювальної техніки та за участю людини, що в керуванні складним об'єктом визначається такими обмеженнями, як обсяг пам'яті й швидкодія при обробці інформації з метою формування чергового керуючого впливу.

Значне зменшення цих обмежень може бути досягнуто за рахунок використання методів штучного інтелекту. Практично необмежений обсяг пам'яті сучасного комп'ютера дає можливість накопичувати великі масиви знань та узагальнювати досвід багатьох експертів щодо керування складним об'єктом, а висока швидкодія комп'ютера дозволяє обробляти вказану інформацію за короткі часові відрізки, забезпечуючи при цьому формування керуючого впливу при керуванні складним об'єктом у реальному часі.

Таким чином, реалізація методів штучного інтелекту в комп'ютеризованій системі керування складним об'єктом надає великі можливості для створення інтелектуальної системи керування, яка буде забезпечувати ефективне керування складним об'єктом за рахунок швидкого використання великих обсягів інформації про знання й навички людини з керування складним об'єктом.

1.2. Експертні системи керування складними об'єктами

Першими серед інтелектуальних систем, в яких здійснювалося накопичення знань людини та обробка цих знань з метою формування (прийняття) рішень, були статичні (традиційні) експертні системи. У цих системах введені знання людини зберігаються незмінними у процесі функціонування системи (не враховуючи набуття нових знань та їх узагальнення).

Структурну схему експертної системи наведено на рис.1.1.

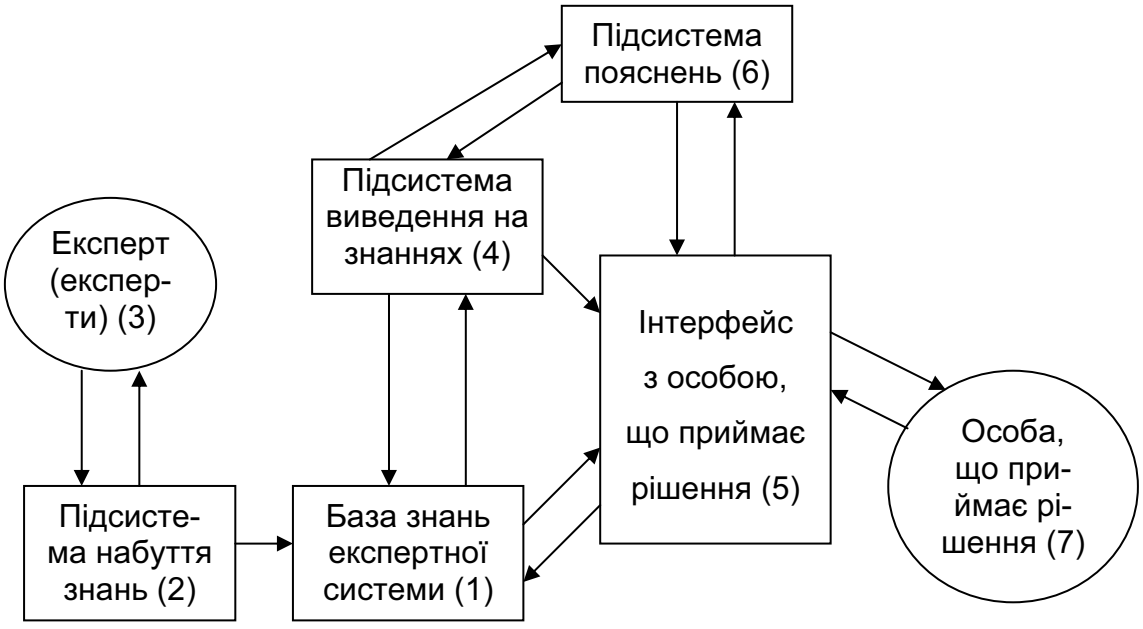


Рис. 1.1. Схема традиційної експертної системи

База знань (1) – компонент експертної системи (підсистеми), призначена для накопичення знань та їх узагальнень на машинних носіях в електронному вигляді; підсистема набуття знань (2) від експерта (3) призначена для формалізації й редагування знань з метою їх подання в електронному вигляді в базі знань (1).

Рішення формується у результаті виведення на знаннях за допомогою підсистеми виведення (4). Сформоване таким чином рішення за допомогою інтерфейсу з особою, що приймає рішення (5), та системи пояснень (6) надається особою, що приймає рішення (ОПР) (7), яка остаточно затверджує отримане рішення, а також надає експертній системі задачу за допомогою інтерфейсу (5).

Статична експертна система може здійснювати накопичування потрібної кількості знань від експертів та обробку й узагальнення цих знань з досить великою швидкістю, яку забезпечує комп'ютер. Такі риси є дуже привабливими для використання експертних систем у керуванні складними об'єктами. Але керування будь яким об'єктом пов'язане з низкою особливостей, які не можуть бути враховані при використанні розглянутої вище експертної системи.

На рис. 1.2 наведено узагальнену структуру системи керування об'єктом. Досягнення мети керування $\boldsymbol{M}$ забезпечується керуючою системою шляхом впливу на об'єкт керування за допомогою змінних керуючого впливу $\overline{\boldsymbol{U}}$, які формуються керуючою системою на основі векторів стану об'єкта керування $\overline{\boldsymbol{Y}}$ та збурюючого впливу $\overline{\boldsymbol{Z}}$ з боку довкілля.

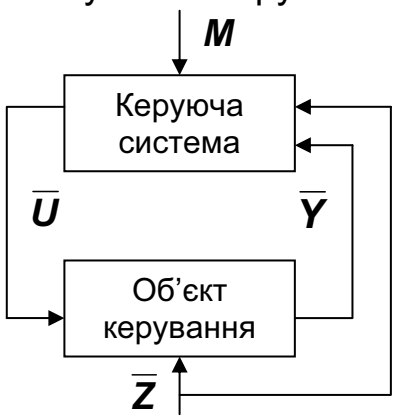


Рис. 1.2. Узагальнена структура системи керування

У процесі функціонування системи керування зміна збурюючого впливу $\overline{\boldsymbol{Z}}$ обумовлює зміну вектора стану об'єкта керування $\overline{\boldsymbol{Y}}$, у результаті цього керуюча система змінює керуючий вплив $\overline{\boldsymbol{U}}$ таким чином, щоб наблизити об'єкт до мети керування $\boldsymbol{M}$.

Порівнюючи схеми на рис. 1.1 і 1.2, бачимо, що застосування експертної системи (рис. 1.1) у системі керування можливе, якщо рішення експертної системи будуть безпосередньо здійснювати керування, тобто забезпечувати наближення до мети керування при різних значеннях збурюючих впливів. При цьому мета керування $\boldsymbol{M}$ має бути задана ОПР (7) за допомогою інтерфейсу (5), тобто результати виведення на знаннях, що становлять послідовність правил продукції (у разі бази знань продукційного типу), має виконувати керуючий вплив на об'єкт керування залежно від збурюючого типу $\overline{\boldsymbol{Z}}$. Таким чином, кожне $\boldsymbol{j}$-те правило продукції як елемент послідовності, що отримана виведенням на знаннях, можна подати у вигляді множин.

Якщо $\{\boldsymbol{P}^{\boldsymbol{a}}_{\boldsymbol{s}_j}(\overline{\boldsymbol{Z}}), \boldsymbol{P}^{\boldsymbol{a}}_{\boldsymbol{q}_j}(\overline{\boldsymbol{Z}}), \ldots, \boldsymbol{P}^{\boldsymbol{a}}_{\boldsymbol{r}_j}(\overline{\boldsymbol{Z}})\}$, тоді $\{\boldsymbol{D}_{\boldsymbol{h}_j}(\tau), \boldsymbol{D}_{\boldsymbol{k}_j}(\tau), \ldots\}$

де $\{\boldsymbol{P}^{\boldsymbol{a}}_{\boldsymbol{s}_j}(\overline{\boldsymbol{Z}}), \boldsymbol{P}^{\boldsymbol{a}}_{\boldsymbol{q}_j}(\overline{\boldsymbol{Z}}), \ldots, \boldsymbol{P}^{\boldsymbol{a}}_{\boldsymbol{r}_j}(\overline{\boldsymbol{Z}})\}$ – предикати антецедента, залежно від збурень $\overline{\boldsymbol{Z}}(\tau)$ у момент часу τ;

$\{D_{h_j}(\tau), D_{k_j}(\tau), \ldots\}$ – керуюча послідовність дій консеквента, після завершення яких будуть задоволені предикати консеквента $\{P_{r_j}, P_{q_j}, \ldots\}$.

Виходячи з цього, узагальнену схему інтелектуальної системи керування з використанням експертної системи можна подати у вигляді, показаному на рис. 1.3.

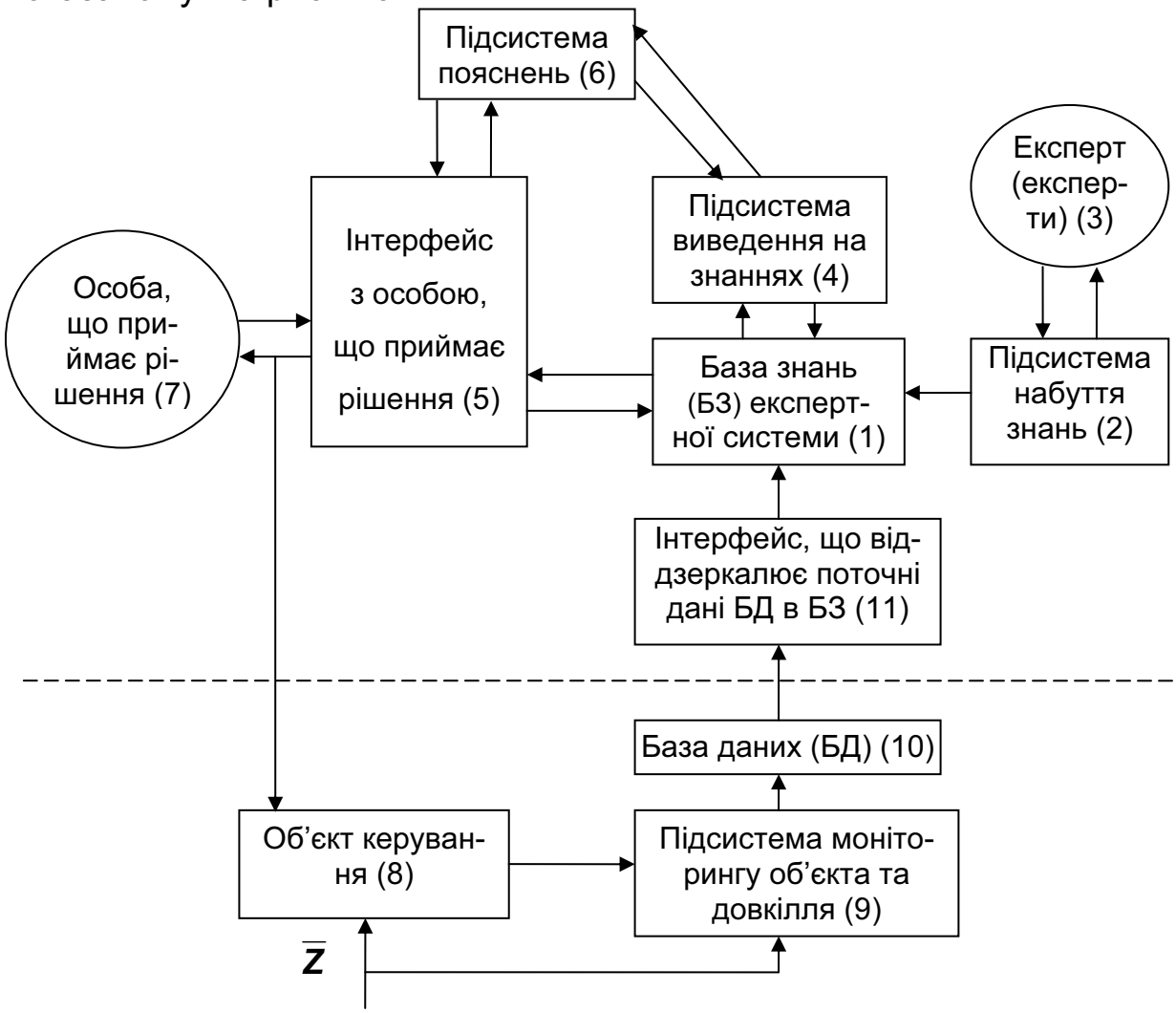


Рис. 1.3. Узагальнена схема інтелектуальної системи керування з використанням ДЕС

З викладеного вище та наведеної схеми видно, що в інтелектуальній системі керування експертна система обов'язково має бути пов'язана з базою поточних даних стану об'єкта та довкілля, тобто має бути динамічною експертною системою (ДЕС).

Можливості інтелектуальної системи керування (ІСК) повинні бути більшими, ніж у системи на рис. 1.2, тому що керування здійснюється не шляхом зміни вектора керування, а шляхом визначення послідовностей керуючих операцій.

Як видно з рис. 1.3, динамічна експертна система керування здійснює процеси керування, які розглянуто на рис. 1.2.

На рис. 1.3, як і на рис. 1.2, відображено визначення поточного стану об’єкта керування (8), але системою моніторингу (9), яка результати вимірів записує до бази даних (10).

Результати вимірів в цьому випадку є вихідними даними для формування керуючого впливу $\overline{U}$ за допомогою експертної системи, що подібна зображеній на рис. 1.1. Таким чином, схему інтелектуальної системи (див. рис. 1.3) можна розділити на об’єкт керування та керуючу систему, як на рис. 1.2. До керуючої підсистеми на рис. 1.3 можна віднести систему моніторингу (9), базу даних (10) та динамічну експертну систему (ДЕС) (відкреслена пунктиром).

Відмітною рисою ДЕС є наявність модуля відображення поточних даних (11) у БД (10) до бази знань (БЗ) динамічної експертної системи. ДЕС формує послідовність керуючих операцій (дій) на об’єкт керування.

1.3. Інтелектуальні інтегровані системи керування

У деяких випадках методи керування в інтелектуальних системах доцільно використовувати разом із традиційними методами керування (системи керування з від’ємним зворотним зв’язком). Такі системи керування називаються інтелектуальними інтегрованими системами керування (ІІСК), які відрізняються від структури ІСК, що наведена на рис. 1.3, модулем традиційних систем керування (12). Цей модуль використовує поточні дані БД (10) як вихідні для реалізації традиційних методів керування, наприклад методу від’ємного зворотного зв’язку в замкнених системах керування.

У цій системі в загальному випадку реалізуються три інформаційних контури зворотного зв'язку з об'єктом: традиційний контур керування, що містить систему моніторингу, БД, засоби реалізації традиційних методів; інтелектуальний контур, що охоплює БД, модуль відображення поточного стану об'єкта в БЗ, ДЕС і ОПР; експертний контур керування, до складу якого входять експерти разом із джерелами інформації (Е), підсистема придбання знань (ППЗ), ДЕС, ОПР і система реалізації керування на об'єкті.

Аналіз схеми ІІСК на рис. 1.4 дає можливість визначити ієрархію з трьох замкнених інформаційних контурів:

1 – традиційного керування;

2 – інтелектуального керування;

3 – експертний контур керування.

У контурі традиційного керування (показано подвійною пунктирною лінією) реалізуються традиційні методи керування (наприклад метод від’ємного зворотного зв’язку). Цей контур є найбільш швидкодійним і займає нижній рівень в ієрархії контурів.

Інформаційний контур інтелектуального керування (пунктирна лінія) здійснює керування на основі виводу на знаннях в БЗ. Це повільний контур.

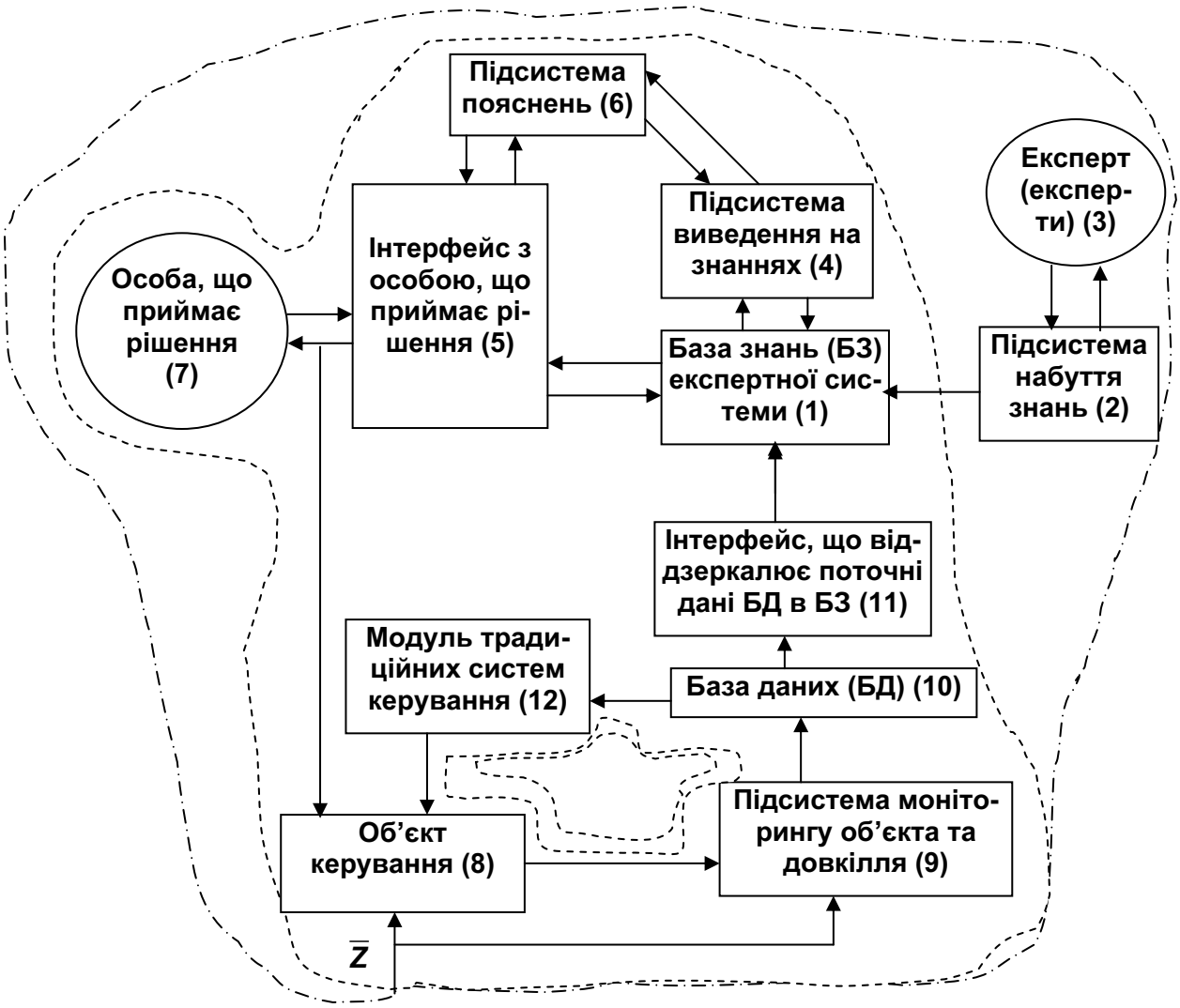


Рис. 1.4. Узагальнена схема ІІСК

Ще більш повільний контур – експертний. Його позначено на рисунку штрихпунктирною лінією. Він починається від джерел нових знань (науково-технічні видання в бібліотеках; знання, здобуті з власного досвіду керування об'єктом тощо), які експерти можуть вводити в систему, підсистему надбання знань, та приєднується до інтелектуального контуру в БЗ.

За ознакою швидкодії ці контури можуть бути упорядковані в ієрархічну структуру, нижній рівень якої займає найбільш швидкий, традиційний контур. Повільніший інтелектуальний контур займає другий рівень. На верхньому ж рівні розташовано найбільш повільно діючий експертний контур, що дозволяє відображати у БЗ наукові досягнення у відповідній галузі, а також узагальнювати власний досвід функціонування інтелектуальної системи керування.

Очевидно, що верхній рівень має можливість коригувати роботу нижнього рівня, оскільки він використовує більший обсяг знань для розв'язання задач у відповідній галузі.

Сумісне функціонування експертного, інтелектуального і традиційного контурів керування дозволяє використовувати всю наявну інформацію для підвищення ефективності роботи об'єкта керування.

Функціонування ІІСК здійснюється таким чином: збурюючий вплив на об'єкт та зміни в довкіллі системою моніторингу вимірюються безпосередньо або через зміну поточного стану об'єкта та відображаються в БД (8). Ця інформація може використовуватися модулем традиційних методів керування для формування керуючих впливів з метою забезпечення цілей керування або віддзеркалюватись у БЗ (1) модулем (11). У результаті виведення на знаннях формуються керуючі впливи у вигляді послідовності керуючих операцій, які за допомогою підсистеми пояснень (6) та інтерфейсу (5) затверджуються й реалізуються ОПР (7) на об'єкті керування (8). Інформаційний потік експертного контуру від експертів (3) через підсистему набуття знань (2) може вносити найновішу інформацію до БЗ (1), яка дозволяє підвищити ефективність роботи інтелектуального контуру ІІСК.

Функціонування зазначених контурів забезпечує ряд переваг розглянутої системи керування:

– високу ефективність прийнятих рішень за рахунок максимального використання наявної в БЗ експертної інформації;

– можливість використання в ДЕС інтуїтивних уявлень експертів у вигляді нечітких знань;

– спільне використання традиційних технологій прийняття рішень й інтелектуальних методів, що забезпечує підвищення швидкодії й ефективності роботи системи.

При створенні ІІСК на основі ДЕС виникає ряд нових проблем, у тому числі:

1) віддзеркалення у БЗ в реальному часі поточного стану об'єкта і навколишнього середовища;

2) скорочення часу виведення на знаннях;

3) керування об'єктом в умовах неповної апріорної інформації про стан об'єкта та довкілля.

Особливе значення має той факт, що алгоритми й методи роботи інтелектуального й експертного контурів у системі мало залежать від області використання ІІСК, тому є можливість створення інструментальних програмних засобів, які придатні для різних сфер застосування.

2. МЕТОДИ ТА АЛГОРИТМИ ФУНКЦІОНУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ КЕРУВАННЯ

2.1. Функціональні підзадачі інтелектуальної системи керування

Порівняльний аналіз функціональних схем статичної (традиційної) експертної системи і ДЕС у складі інтелектуальної системи керування (рис. 1.1, 1.3) свідчить, що процес функціонування ДЕС передбачає реалізацію низки специфічних задач, не притаманних статичній експертній системі. Ці підзадачі загалом складають задачу урахування в реальному часі поведінки об'єкта керування та формування й реалізації рішень у темпі, що задається динамікою змін його стану. Однією з найважливіших є підзадача відображення в базі знань ДЕС поточного стану об'єкта керування й навколишнього середовища (довкілля). Поточний стан довкілля, в якому функціонує ІСК, як правило, змінюється випадково, бо залежить від багатьох випадкових чинників. Поточний стан об'єкта теж змінюється випадково, оскільки він залежить як від впливу збурюючих факторів навколишнього середовища, так і від керуючого впливу з боку ІСК.

При формуванні керуючого впливу шляхом виведення на знаннях з використанням продукційної БЗ необхідно, щоб змінні в предикатах антецедентів правил продукції відповідали значенням останніх вимірів системи моніторингу. До наступного виміру змінних у системі моніторингу повинно бути закінчено формування керуючих операцій на основі виведення на знаннях. У цьому разі в момент наступного виміру результати попереднього вже будуть враховані у сформованому рішенні та керуючі дії будуть запущені (тобто відображення у БЗ поточних даних попереднього виміру вже здійсниться до моменту наступного).

Таким чином, на часовому інтервалі між сусідніми вимірами системи моніторингу повинно бути здійснено відображення у БЗ результатів попереднього виміру, формування послідовності керуючих операцій (з урахуванням результатів попереднього виміру) шляхом виведення на знаннях та має початись виконання сформованої послідовності керуючих операцій. На сусідньому інтервалі після наступного виміру дії, що описані вище, будуть здійснюватись після кожного виміру. На рис. 2.1 зображено (описано) послідовності на двох сусідніх інтервалах між вимірами системи моніторингу. Як видно, час формування послідовності керуючих операцій шляхом виведення на знаннях не може перевищувати період часу між сусідніми замірами. З цього випливає друга підзадача функціонування ІСК – підзадача скорочення часу виведення на знаннях. Розв'язання цієї задачі пов'язане з динамічною декомпозицією БЗ на підмножини правил продукції в кожний момент реального часу функціонування ІСК.

Сукупність підмножин правил продукції бази знань ІСК визначає поточний стан БЗ. При цьому виведення на знаннях здійснюється не на всій БЗ, а тільки на одній підмножині зазначеної сукупності, що дозволяє різко скоротити кількість правил продукцій, які розглядаються при формуванні послідовності керуючих дій шляхом виведення на знаннях після кожного виміру, що здійснюється системою моніторингу.

2.2. Віддзеркалення поточного стану складного об'єкта керування в базі знань інтелектуальної системи

Однією з найважливіших із вказаних проблем є проблема віддзеркалення поточного стану об'єкта у БЗ, оскільки від цього залежить відповідність рішень, що приймаються, реальній ситуації на об'єкті. Ця проблема вирішується шляхом створення спеціального інтерфейсу між БД, даних моніторингу і БЗ, в якій містяться знання експертів. Найбільш зручною є БЗ продукційного типу, оскільки подання знань тут найбільш відповідає способу уявлення знань людиною та правила продукції дозволяють зручно накопичувати знання в процесі функціонування системи [1].

У разі використання продукційної БЗ інтерфейс між БД і БЗ доцільно будувати на основі сукупності метаправил, які віддзеркалюють, з одного боку, склад БЗ, що наповнюється системою моніторингу, а з другого – структуру продукційної БЗ, що визначається змінними, які входять до складу предикатів антецедентів правил продукції БЗ.

У подальшому будемо використовувати такі позначення:

- Ω – множина усіх правил продукції БЗ разом із метаправилами, $\Omega = \{\omega_r\}$, $r = [1, v]$, де v – кількість правил продукції;
- Y – множина змінних, значення яких у БД змінюються в процесі безперервного моніторингу об'єкта;
- Y^* – множина змінних, що входять до складу предикатів антецедентів правил продукції БЗ $Y^* \subseteq Y$;
- P – множина предикатів, що входять до антецедентів і консеквентів правил продукції з множини Ω;
- P_r – множина усіх предикатів, що входять до складу антецедента $\left\{p_{rs}^{(a)}\right\}$, $s \in v_r^{(a)}$ та консеквента $\left\{p_{rh}^{(k)}\right\}$, $h \in v_r^{(k)}$ правила ω_r.

Тут $v_r^{(a)}$, $v_r^{(k)}$ – множини індексів предикатів відповідно антецедента й консеквента правила ω_r.

Визначення 2.1. Правило $\omega_i^{(1)}, i = [v+1, m]$ називається метаправилом першого роду (МПР), якщо антецедент цього правила складається з такого предикату:

$$p_{rs}^{(a)}\left(\left|\tilde{y}_i(t_n) - \tilde{y}_i(t_{n-1})\right| \geq \delta_i\right), \qquad (2.1)$$

де $\tilde{y}_i(t_n)$ – значення змінної $y_i, y_i \in Y^*$ у поточний момент заміру t_n; δ_i – порогова оцінка приросту y_i на інтервалі часу $[t_n - t_{n-1}]$, а кон-

секвент МПР $\omega_i^{(1)}$ містить предикат вигляду

$$p_{ih}^{(k)}\left(y_i = \tilde{y}_i(t_n)\right) = 1, \qquad (2.2)$$

що свідчить про занесення нового значення $\tilde{y}_i(t_n)$ у БД в момент часу t_n.

Значення порогової оцінки δ_i щодо кожної змінної y_i визначається згідно з припустимою похибкою вимірювального каналу, а також значущистю даного параметра і заноситься в предикати антецедентів відповідного МПР при налагодженні інтерфейсу між БД та БЗ для конкретної системи.

Зрозуміло, що для конкретної БЗ кількість МПР у БЗ відповідає кількості змінних Y^*, які входять до предикатів антецедентів БЗ $\left|\Omega^{(1)}\right| = |Y^*|, \Omega^{(1)} \subset \Omega.$

Як видно з (2.1), МПР $\omega_i^{(1)}$ визначає факт суттєвої для БЗ ДЕС зміни значення y_i, що контролюється системою моніторингу.

Визначення 2.2. Віддзеркалення БД у БЗ продукційного типу – це процес обчислення значень предикатів антецедентів $\left\{p_{rs}^{(a)}\right\}, s \in v_r^{(a)}$, для кожного з правил $\omega_r \in \Omega$, які залежать від значення змінної $\tilde{y}_i(t_n), i = [1, m]$, що було виміряне в момент реального часу t_n.

Як видно з цього визначення, процес віддзеркалення БД у БЗ забезпечує відповідність значень предикатів антецедентів продукційних правил поточному стану об'єкта, чим досягається адекватність рішень, що отримані шляхом виведення на знаннях.

Для реалізації процесу віддзеркалення БД у БЗ використовуються метаправила другого роду (МДР).

Визначення 2.3. Продукційне правило $\omega_i^{(2)}$ називається метаправилом другого роду (МДР), якщо антецедент цього правила є предикатом виду $p_j^{(a)}\left(y_i = \tilde{y}_i(t_n)\right) = 1$ і співпадає з консеквентом відповідного МПР $\omega_i^{(1)}$, а консеквент МДР $\omega_i^{(2)}$ визначається умовою виду

$$\forall p_g^{(k)}\left(y_i = \tilde{y}_i(t_n)\right) = 1, \qquad (2.3)$$

де кожний предикат $p_g^{(k)}$ відповідає певному предикату антецедентів правил продукції БЗ, що залежить від $y_i,\ g \in G_i$.

Для даної БЗ будь-якому МПР $\omega_i^{(1)} \in \Omega^{(1)}$ має відповідати своє МДР $\omega_i^{(2)} \in \Omega^{(2)}$.

Як видно з визначень 2.1, 2.3, доцільно поєднати метаправила $\omega_i^{(1)}, \omega_i^{(2)}$ у єдиному, узагальненому метаправилі.

Визначення 2.4. Метаправило $\omega_i^{(u)}$ називається узагальненим метаправилом (УМП), якщо його антецедент збігається з антецедентом

МПР $\omega_i^{(1)}$, а консеквент – із консеквентом МДР $\omega_i^{(2)}$.

Таким чином, викладений вище підхід до побудови інтерфейсу між БД та БЗ з використанням метаправил надає такі можливості: по-перше, безперервно віддзеркалювати, майже без запізнювання, у БЗ поточний стан об'єкта керування з метою забезпечення функціонування ДЕС у межах ІІСК у реальному часі, оскільки віддзеркалення значень змінних БД у БЗ здійснюється практично в моменти кожної суттєвої зміни значень відповідних параметрів; по-друге, побудувати уніфікований інтерфейс між БД і БЗ, який може бути адаптований до великого класу складних об'єктів.

Під впливом зазначених вище контурів керування в процесі функціонування ДЕС серед множини правил $\Omega = \{\omega_r\}, r = \overline{1,V}$, продукційної БЗ утворяться такі підмножини: $\Omega^{(K)}$ – правила конфліктного набору; $\tilde{\Omega}^{(K)}$ – правила-претенденти на включення до конфліктного набору; $\Omega^{(B)}$ – правила, що виконуються в даний момент; $\Omega^{(З)}$ – правила продукції, операції яких завершилися в сучасний момент; $\Omega^{(НВ)}$ – правила продукції, операції яких не виконані в сучасний момент у межах припустимого часового інтервалу; $\Omega^{(БЛ)}$ – правила, що заблоковані на встановлений час для попередження їхнього повторного спрацьовування.

Припустима тривалість виконання операцій, яка обумовлена експертами, в правилах підмножини $\Omega^{(НВ)}$ на даний момент може бути перевищена, зокрема, через несправності елементів системи.

У процесі функціонування ДЕС склад елементів зазначених вище підмножин постійно змінюється таким чином, що в будь-який t_n-й момент у БЗ ДЕС можна виділити дві підмножини:

$$\hat{\Omega}_n = \Omega_n^{(K)} \cup \tilde{\Omega}_n^{(K)} \cup \Omega_n^{(B)} \cup \Omega_n^{(З)} \cup \Omega_n^{(НВ)} \cup \Omega_n^{(БЛ)}, \Omega_n^* = \Omega / \hat{\Omega}_n .$$

Підмножину $\hat{\Omega}_n$ доречно назвати активною частиною БЗ ДЕС у момент t_n, а Ω_n^* – пасивною частиною БЗ у t_n-й момент часу.

Підмножини $\Omega_n^{(K)}, \tilde{\Omega}_n^{(K)}, \Omega_n^{(B)}, \Omega_n^{(З)}, \Omega_n^{(НВ)}, \Omega_n^{(БЛ)}$ та Ω_n^* характеризують статус кожного правила БЗ ДЕС у поточний t_n-й момент функціонування ДЭСУ й, отже, стан БЗ у цілому.

Визначення 2.5. Поточний стан продукційної БЗ у t_n-й момент функціонування ДЕС визначається складом підмножин активної частини $\hat{\Omega}_n$ БЗ, що залежить від поточного стану об'єкта й наявної експертної інформації.

Роль кожної з зазначених підмножин полягає у прискоренні процесу виведення на знаннях у ДЕС завдяки зменшенню кількості правил продукцій, що розглядаються у кожний момент часу, а також у виявленні відхилень у роботі системи й об'єкта керування (у результаті формування підмножини $\Omega^{(НВ)}$).

Підмножина конфліктного набору $\Omega^{(K)}$ дозволяє за допомогою виведення на знаннях сформувати й реалізувати розв'язок на об'єкті у вигляді послідовності операцій, що запускаються безпосередньо в результаті виміру в поточний момент t_n змінних на об'єкті.

Підмножина правил-претендентів на включення до конфліктного набору $\tilde{\Omega}^{(K)}$ служить для швидкого формування конфліктного набору $\Omega^{(K)}$ у поточний момент виміру параметрів об'єкта системою моніторингу.

Склад елементів підмножини $\Omega^{(B)}$ правил, що виконуються, визначається всіма операціями, реалізованими на об'єкті в поточний момент часу.

Підмножина $\Omega^{(З)}$ завершених у сучасний момент часу операцій виділяється з $\Omega^{(B)}$ і може бути використана у підсистемі пояснень ДЕС для визначення шляху формування рішень.

Підмножина $\Omega^{(B)}$ містить у своєму складі підмножину $\Omega^{(НВ)}$ правил, що не виконалися, тобто тих правил, для яких відповідні операції не завершилися в межах припустимого часового інтервалу (заданого апріорно експертами) внаслідок, наприклад, порушень у роботі системи.

Результат виміру $y_i(t_n)$ системою моніторингу в поточний момент часу t_n будь-якого параметра $y_i \in Y, i = \overline{1,n}$, об'єкта може викликати в загальному випадку зміну значень предикатів, що залежать від цього параметра, $P_m(y_i,...) \in P, m \in V_m$, і, отже, зміни в антецедентах $\{p^{(а)}\}$ і консеквентах $\{p^{(к)}\}$ правил БЗ, куди входять дані предикати; тут V_m – множина індексів предикатів, що залежать від y_i.

Сформулюємо умови включення й вилучення правил продукцій для описаних вище підмножин БЗ безпосередньо після виміру параметрів у t_n-й момент функціонування ДЕС.

Очевидно, у $\Omega_n^{(K)}$ включаються правила продукції, усі предикати антецедентів яких задовольняються, тобто

$$\omega_p \in \Omega_n^{(K)} \Big| (\forall p_\chi \in \{p_{p\alpha}^{(a)}\}) \Big| p_\chi = 1, \alpha \in \upsilon_p^{(a)}, \qquad (2.4)$$

де $\upsilon_p^{(a)}$ – множина предикатів антецедента правила ω_P.

У конфліктний набір $\Omega_n^{(K)}$ можуть бути включені правила продукції тільки з підмножини $\tilde{\Omega}^{(K)}$, оскільки умова задовільнення усіх предикатів антецедента (достатня умова) містить умову задовільнення у даний момент часу t_n будь-якої частини множини предикатів антецедента (необхідна умова).

Підмножина правил-претендентів $\tilde{\Omega}_n^{(K)}$ на включення до конфліктного набору $\tilde{\Omega}_n^{(K)}$ формується з умови, що в антецедентах цих правил існує хоча б один предикат, що змінив своє значення з нуля на одиницю в результаті останнього виміру в момент часу t_n:

$$\omega_m \in \tilde{\Omega}_n^{(K)} \Big| (\exists p_\beta(y_i,...) \in \{p_{m\Gamma}^{(a)}\}) \wedge (p_\beta(y_i,...) = 1), \Gamma \in \upsilon_m^{(a)}, \quad (2.5)$$

де $\upsilon_m^{(a)}$ – множина індексів предикатів антецедента правил ω_m.

У випадку, якщо в результаті виміру в поточний момент t_n значення змінної $y_i(t_n)$ усі предикати антецедента не будуть задовільнені, тоді відповідні правила продукції повинні бути виключені з підмножини $\tilde{\Omega}_n^{(K)}$ безпосередньо після виміру (до початку наступного виміру).

До підмножини $\Omega_n^{(B)}$ правил, що виконуються у поточний момент часу t_n, включаються правила $\Omega_u^{(K)}$ тільки з конфліктного набору (в порядку, обумовленому схемою й стратегією виведення на знаннях), $\Omega_n^{(K)} \subset \Omega_n^{(B)}$; у $\Omega_n^{(B)}$ також містяться усі раніше запущені на виконання правила, для яких справедливо

$$\begin{aligned} &\omega_j \in \Omega_n^{(B)} \Big| (P(\delta_j - \tau_j \geq 0) = 1) \wedge (\exists p_\gamma \in \{p_{j\sigma}^{(K)}\} \Big| p_\gamma = 0) \\ &\tau_j = t_n - t_j, \sigma \in \upsilon_j^{(K)}, \end{aligned} \quad (2.6)$$

де $\delta_j \in \psi, j = \overline{1,V}$ – встановлена тривалість реалізації операції, що відповідає правилу ω_j;

t_j – момент запуску на виконання правила ω_j;

$\upsilon_j^{(K)}$ – множина індексів предикатів консеквента ω_j.

Завершені правила з $\Omega_n^{(3)}, \Omega_n^{(3)} \subset \Omega_n^{(B)}$, для яких виконана умова

$$\omega_l \in \Omega_n^{(3)} \Big| (\forall P_\eta \in \{p_{l\rho}^{(K)}\} \Big| P_\eta = 1), \rho \in \upsilon_l^{(K)}, \quad (2.7)$$

де $\upsilon_l^{(K)}$ – множина індексів предикатів консеквента в правила ω_l, вилучаються з підмножини $\Omega_n^{(B)}$ і включаються в $\Omega_n^{(БЛ)}$, тобто предикати консеквентів цих правил виявилися виконаними за даними виміру на об'єкті, проведеного після виконання відповідної операції.

Правила типу $\omega_\xi \in \Omega^{(3)}$, що вже завершені, для запобігання їхнього повторного спрацьовування (якщо антецеденти цих правил були колись задовільнені) повинні бути заблоковані на визначений експертом або передбачений регламентом час $\{\lambda_r\} = \Lambda, r = \overline{1,V}$, тому в t_n-й момент часу в $\Omega_n^{(БЛ)}$ необхідно включити правила з $\Omega_n^{(3)}$ й усі раніше завершені правила, для яких

$$\omega_\xi \in \Omega_n^{(БЛ)} \Big| (P(\lambda_\xi - \bar{\tau}_\xi \geq 0) = 1), \bar{\tau}_\xi = t_u - \bar{t}_\xi, \quad (2.8)$$

де $\bar{t}_\xi$ – момент завершення операції, що відповідає правилу ω_ξ.

Підмножина $\Omega_n^{(НВ)}$ правил продукції, що не виконуються на припустимому тимчасовому $\Omega_n^{(B)}$ інтервалі, $\Omega_n^{(НВ)} \subset \Omega_n^{(B)}$, доцільно виділяти з підмножини з метою використання цих правил для аналізу причин порушень у роботі об'єкта.

У $\Omega_n^{(НВ)}$ включаються правила з незадоволеним консеквентом, для яких тривалість реалізації δ_q у поточний момент t_n виявилася перевищеною, тобто

$$\omega_q \in \Omega_n^{(НВ)} \Big| (P(\delta_q - \tau_q \geq 0) = 0) \wedge (\exists p \in \{p_{q\theta}^{(К)}\} | p = 0), \quad \tau_q = t_n - t_q, \Theta \in \upsilon_q^{(К)}, \tag{2.9}$$

де $\upsilon_q^{(К)}$ – множина індексів предикатів консеквента правила ω_q;

t_q – момент початку операції, що відповідає правилу ω_q.

Очевидно, правила $\hat{\Omega}_n$ активної частини БЗ, що не задовольняють у момент t_u жодному з умов (2.1)–(2.6), включаються у пасивну частину Ω_n^*.

Для забезпечення нормальної роботи ДЕС принципово важливо, щоб час Δ_u формування підмножин активної частини $\hat{\Omega}_n$ в будь-який t_n-й момент функціонування системи був меншим за інтервал між вимірами параметрів об'єкта системою моніторингу:

$$[t_{n+1} - t_n]_{min} > \Delta_n. \tag{2.10}$$

Процес формування підмножин активної частини відбувається на основі принципу динамічної декомпозиції БЗ.

Визначення 2.6. Динамічна декомпозиція БЗ ДЕС у момент часу t_n – це процес формування множин на основі переходу правил з однієї підмножини в іншу в рамках

$$\Omega_n^{(К)} \cup \tilde{\Omega}_n^{(К)} \cup \Omega_n^{(В)} \cup \Omega_n^{(З)} \cup \Omega_n^{(НВ)} \cup \Omega_n^{(БЛ)} \cup \Omega^* \cup \Omega$$

за умовами (2.4)–(2.10) за час, що задовольняє умові (2.10).

Приклад 2.1.

Для ілюстрації процесу декомпозиції розглянемо в момент часу t_{n+1} поточний стан БЗ ДЕС, що включає десять правил $\{\omega_r\} = \Omega, r = \overline{1,10}$, з яких два є такими, що виконуються, $\Omega_n^{(В)} = \{\omega_4, \omega_8\}$.

Припустимо, що в t_{n+1}-й момент системою моніторингу отримано нове значення $y_i(t_{n+1})$ параметра $y_i \in Y$, від якого залежать предикати антецедентів правил ω_1, ω_3 таким чином, що в результаті зміни значень цих предикатів антецедент правила ω_1 виявився цілком задовільненим, а в антецеденті правила ω_3 залишилися предикати, що мають нульове значення.

Тоді поточний стан БЗ безпосередньо після $n+1$-го моменту часу (в момент $t_{n+1} + \Delta_{n+1}$, де $\Delta_{n+1} << t_{n+2} - t_{n+1}$) за умовами (2.4)–(2.10) буде

визначатися набором таких підмножин:

$$\tilde{\Omega}_{n+1}^{(K)} = \{\omega_1, \omega_3\}; \Omega_{n+1}^{(K)} = \{\omega_1\}; \omega_3 \notin \Omega_{n+1}^{(K)};$$

$$\Omega_{n+1}^{(B)} = \{\omega_1\}; \Omega_{n+1}^{(З)} = \{\omega_8\}; \Omega_{n+1}^{(БЛ)} = \{\omega_8\}; \Omega^{(З)} \cap \Omega^{(БЛ)} \neq 0;$$

$$\Omega_{n+1}^{(НВ)} = \{\omega_4\}; \ \Omega^{(НВ)} \subseteq \Omega^{(B)};$$

$$\Omega_{n+1}^{*} = \{\omega_2, \omega_3, \omega_5, \omega_6, \omega_7, \omega_9, \omega_{10}\}.$$

Формування наведених вище підмножин відбувається на часовому інтервалі $t_{n+1} < \tau << t_{n+2}$.

Будемо вважати, що всі предикати антецедентів правил ω_9, ω_{10}, а також консеквента ω_1 залежать від значення $y_j(t_{n+2})$ параметра виміру $y_j \in Y, j = i,$ що змінився в момент t_{n+2}. При цьому поточний стан $t_{n+2} + \Delta_{n+2}$, де $\Delta_{n+2} << t_{n+3} - t_{n+2}$, БЗ у момент t_n визначається такими підмножинами:

$$\tilde{\Omega}_{n+2}^{(K)} = \{\omega_9, \omega_{10}\}; \Omega_{n+2}^{(K)} = \{\omega_9, \omega_{10}\}; \Omega_{n+2}^{(B)} = \{\omega_9, \omega_{10}\};$$

$$\Omega_{n+2}^{(З)} = \Omega_{n+2}^{(БЛ)} = \{\omega_1\}; \Omega_{n+2}^{(НВ)} = \{\omega_4\};$$

$$\Omega_{n+2}^{*} = \{\omega_2, \omega_3, \omega_5, \omega_6, \omega_7, \omega_8\}.$$

Для формування поточного стану БЗ ДЕС після вимірів у кожний із моментів часу t_{n+1} і t_{n+2} було досить перевірити лише два правила (ω_1, ω_3 і ω_9, ω_{10} відповідно), що в п'ять разів менше загальної кількості правил у БЗ.

Скорочення кількості правил можливе за рахунок того, що формування конфліктного набору відбувається тільки в результаті обчислення значень предикатів в антецедентах правил-претендентів, тобто правил, у яких в результаті виміру хоча б один предикат антецедента змінив своє значення з 0 на 1.

Таким чином, наведений вище підхід до функціонування БЗ ДЕСК на основі динамічної декомпозиції дозволяє істотно скоротити кількість правил продукцій, що перевіряються на кожному такті роботи системи, а також за рахунок формування конфліктного набору на основі множини правил претендентів прискорити процес виведення на знаннях.

2.3. Стратегія формування рішень за умови неповної апріорної інформації про стан складного об'єкта керування

Основою формування керуючих впливів у ДЕС є інформація про поточний стан об'єкта в даний момент часу та експертна інформація, що визначається рівнем наукових досягнень у даній галузі, досвідом експлуатації системи й іншої інформації, що була отримана від експертів [2, 3].

Керуючий вплив у ДЕС може бути визначений на основі трьох різних стратегій виведення на знаннях у продукційної БЗ: прямим, зворотним і двоспрямованим виведенням [1].

Кожна з цих стратегій має свої особливості, що істотно впливають на ефективність функціонування ДЕС. Результатом прямого виведен-

ня на знаннях у БЗ є послідовність керуючих операцій $\{ D(\tau) \}$, $\tau \in T$, $v \in V$, кожна з яких починає виконуватися у визначений момент реального часу τ на часовому інтервалі T розв'язання задачі для досягнення цілі функціонування системи (виконання цільових предикатів). Особливість цієї стратегії полягає в тому, що в загальному випадку в сформовані розв'язки можуть бути включені керуючі операції, які зовсім не пов'язані з цільовими предикатами. Крім того, в міру віддалення у майбутнє від сучасного моменту ухвалення розв'язку τ^* спрямованість прямого виведення на задоволення цільових предикатів погіршується додатково внаслідок похибок прогнозування випадкових обурень, що впливають на результат виведення на знаннях. Тому рішення, отримане прямим виведенням на знаннях може у загальному випадку містити надлишкові керуючі операції, виконання яких не є обов'язковим для досягнення цілей функціонування системи [4].

Істотним є і те, що операції, запропоновані в сформованому рішенні для майбутніх моментів часу, віддалених від поточного τ^* (при якому формувалося рішення), можуть бути взагалі помилковими через похибки прогнозування випадкових обурень.

Таким чином, стратегія прямого виведення в загальному випадку може передбачати надлишкову кількість керуючих операцій на об'єкті й тому на віддалених від τ^*-го моменту часу можливі помилкові дії. Цього недоліку позбавлена стратегія зворотного виведення, при якій процес починається від цільового предиката, і тому одержаний розв'язок містить тільки керуючі операції, що зв'язані з цільовими предикатами. Однак сформований розв'язок залежить від похибок прогнозування випадкових збурних діянь $\bar{Z}(\tau)$ на всьому інтервалі від поточного моменту часу τ^* до моменту досягнення цілі $\tau^{(ц)}$.

Множину дій $D^{Г}_{\tau^*}$, виконання яких на часовому інтервалі $[\tau^*, \tau^{(ц)}]$ гарантує досягнення цілі, можна приблизно оцінити об'єднанням підмножин:

$$D^{Г}_{\tau^*} \cong \bigcup_{\eta \in H} \{D_v(\bar{Z}^{П}_{\eta}(\tau)\}, \tau \in [\tau^*, \tau^{(ц)}], v \in V, \qquad (2.11)$$

де $\bar{Z}^{П}_{\eta}(\tau)$ – одна з можливих прогнозованих реалізацій випадкового процесу $\bar{Z}(\tau)$ на інтервалі $[\tau^*, \tau^{(ц)}]$;

H – досить представницька підмножина порядкових індексів, прогнозованих реалізацією випадкового процесу на інтервалі $[\tau^*, \tau^{(ц)}]$ у поточний момент часу τ^*;

$D_v(\bar{Z}^{П}_{\eta}(\tau))$ – дії, що виконуються з моменту τ при $\bar{Z}^{П}_{\eta}$-й прогнозованій реалізації випадкового збурення;

V – множина порядкових індексів дій.

На основі викладеного доцільним є механізм двоспрямованого виведення, при якому пряме виведення реалізується для множини $D^{Г}_{\tau^*}$,

при цьому отримане рішення реалізується на об'єкті тільки на інтервалі $[\tau^*, \tau^* + \Delta]$, де Δ – часовий інтервал, на якому є досить повна апріорна інформація про об'єкт керування.

Після виконання керуючих операцій у момент часу $\tau^* + \Delta$ визначається більш точна оцінка збурень $\bar{Z}(\tau)$ на інтервалі $[\tau^* + \Delta, \tau^{ц}]$ для якої зворотним виведенням знов формується розв'язок за формулою (2.11), тобто визначається множина $D^{Г}_{\tau^*+\Delta}$ для моменту часу $\tau^* + \Delta$. На правилах продукцій, що відповідають цій множині, знову здійснюється пряме виведення на знаннях, результати якого у вигляді керуючих впливів реалізуються на об'єкті на часовому інтервалі $[\tau^* + \Delta, \tau^* + 2\Delta]$. Далі процес повторюється.

Для отриманої уточненої оцінки обурень на інтервалі $[\tau^* + \Delta, \tau^{ц}]$ зворотним виводом знову формується розв'язок за формулою (2.11) у вигляді гарантованої послідовності дій $D^{Г}_{\tau^*+\Delta}$ реалізованої на часовому інтервалі $[\tau^* + \Delta, \tau^* + 2\Delta]$ і т.д.

Таким чином, механізм двоспрямованого виведення реалізується у вигляді ітераційного процесу, на кожній ітерації якого зворотним виводом визначається множина $D^{Г}_{\tau^*+\alpha\Delta}$, що гарантує досягнення мети при його виконанні на інтервалі часу $[\tau^* + \alpha\Delta, \tau^{ц}]$ і формується прямим виведенням раціональне рішення у вигляді послідовності дій на об'єкті, що виконуються на інтервалі реального часу $[\tau^* + \alpha\Delta, \tau^* + (\alpha + 1)\Delta]$.

Використання механізму ітераційного двоспрямованого виведення надає можливість зменшити кількість розглянутих правил продукцій (безпосередньо діючих на об'єкт) при гарантованому досягненні мети з обліком похибок прогнозування збурень (параметричних і координатних), що діють на об'єкт керування.

Комплексне вирішення розглянутих проблем дозволяє створювати ефективні ІІСК складними об'єктами в реальному часі.

Специфіка стратегій виведення на знаннях (ВНЗ) (прямої і зворотної) не дозволяє безпосередньо використовувати їх при керуванні об'єктом з неповною інформацією в реальному часі.

У зв'язку з цим при керуванні складним об'єктом в екстремальному режимі доцільно застосувати особливу стратегію ВНЗ, що сполучає переваги прямого і зворотного висновку.

Алгоритм, що реалізує цю процедуру, включає такі кроки:

1. Прогноз значення цільового предиката і моменту його досягнення $\tau^{(ц)}$.

2. Прогноз на інтервалі $[\tau^*, \tau^{(ц)}]$ збурень $\bar{Z}^{П}(\tau^*)$.

3. Формування на інтервалі $[\tau^*, \tau^{(ц)}]$ зворотною стратегією ВНЗ множини безлічі дій $D^{Г}_{\tau^*}$ для прогнозованих збурень.

4. Визначення тимчасового інтервалу Δ, на якому існує досить повна інформація про збурення.

5. Формування за допомогою прямої стратегії ВНЗ послідовності дій $\hat{D}_{\tau^*}$, $\hat{D}_{\tau^*} \supset D^{\Gamma}_{\tau^*}$, сумарна тривалість яких не перевищує Δ.

6. Реалізація дій з множини $\hat{D}_{\tau^*}$ на об'єкті.

7. Якщо цільовий предикат задовольняється – закінчення роботи, в іншому випадку – перехід до кроку 1 при значенні поточного моменту часу $\tau^* + \Delta$.

Приклад 2.2.

Розглянемо частину ІІСППР ліквідації лісових пожеж, яка містить фрагмент БД, що складається зі значень предикатів ($P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_{10}, P_{11}, P_{12}, P_{13}, P_{14}, P_{15}, P_{16}, P_{17}, P_{18}, P_{19}, P_{20}$) і дій ($D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8, D_9, D_{10}$); систему моніторингу, що відслідковує швидкість і напрямок вітру в осередку пожежі (ОП), а також фрагмент БЗ, що складається з п'ятнадцяти правил продукцій:

1. ЯКЩО надійшов сигнал про пожежу у південній частині зони об'єкта ТА вітер північний, ТО визначити швидкість вітру в зоні ОП, ТОДІ швидкість вітру в зоні ОП визначено.

2. ЯКЩО швидкість вітру в зоні ОП визначено ТА швидкість вітру в зоні ОП $V_в < 5$ м/с, ТО визначити швидкість просування крайки ОП по фронту, ТОДІ швидкість просування крайки ОП по фронту $V_п < 10$ м/хв*.

3. ЯКЩО швидкість вітру в зоні ОП визначено ТА швидкість вітру в зоні ОП $5 \le V_в \le 10$ м/с, ТО визначити швидкість просування крайки ОП по фронту, ТОДІ швидкість просування крайки ОП по фронту $10 \le V_п < 15$ м/хв.

4. ЯКЩО швидкість вітру в зоні ОП визначено ТА швидкість вітру в зоні ОП $V_в > 10$ м/с, ТО визначити швидкість просування крайки ОП по фронту, ТОДІ швидкість просування крайки ОП по фронту $V_п > 15$ м/хв.

5. ЯКЩО швидкість просування крайки ОП по фронту $V_п < 10$ м/хв, ТО ідентифікувати тип пожежі, ТОДІ пожежа низова середня.

6. ЯКЩО швидкість просування крайки ОП по фронту $10 \le V_п < 15$ м/хв, ТО ідентифікувати тип пожежі, ТОДІ пожежа низова сильна.

7. ЯКЩО швидкість просування крайки ОП по фронту $V_п > 15$ м/хв, ТО ідентифікувати тип пожежі, ТОДІ пожежа верхова ТА об'єкт у небезпеці.

8. ЯКЩО пожежа низова середня, ТО направити ресурси для гасіння до ОП, ТОДІ ресурси для гасіння розташовані біля ОП.

9. ЯКЩО пожежа низова сильна, ТО розподілити і спрямувати ресурси для гасіння до південної межі об'єкта й уздовж перешкод по

* В лісовій пірології швидкість розповсюдження лісової пожежі вимірюється в м/хв.

флангах ОП, ТОДІ ресурси для гасіння розподілені й розташовані на південній межі об'єкта й уздовж перешкод по флангах ОП.

10. ЯКЩО пожежа верхова, ТО всі наявні ресурси для гасіння зосередити біля південної межі об'єкта, ТОДІ ресурси для гасіння розгорнуті на південній межі об'єкта.

11. ЯКЩО об'єкт у небезпеці, ТО почати евакуацію людей і рухомого майна з об'єкта, ТОДІ люди і рухоме майно з об'єкта евакуйовані.

12. ЯКЩО ресурси для гасіння розташовані біля ОП, ТО віддати наказ про зупинку пожежі, ТОДІ пожежа зупинена.

13. ЯКЩО ресурси для гасіння розподілені й розташовані на південній межі об'єкта й уздовж перешкод по флангах ОП, ТО провести охоплення зони ОП по фронту й флангах, ТОДІ охоплення зони ОП по фронту й флангах проведене.

14. ЯКЩО охоплення зони ОП по фронту й флангах проведене, ТО зупинити пожежу, ТОДІ пожежа зупинена.

15. ЯКЩО об'єкт у небезпеці ТА ресурси для гасіння розгорнуті на південній межі об'єкта ТА люди й рухоме майно з об'єкта евакуйовані, ТО створити опорну смугу для пуску зустрічного вогню, ТОДІ опорна смуга для пуску зустрічного вогню створена.

Значення “істина” предикатів з множини $\{P_i\}$, $i = \overline{1,20}$ свідчить про виникнення в процесі розвитку й ліквідації лісової пожежі таких умов:

P_1 – надійшов сигнал про пожежу в південній частині зони об'єкта;

P_2 – вітер північний;

P_3 – швидкість вітру в зоні ОП визначена;

P_4 – швидкість вітру в зоні ОП $V_в < 5$ м/с;

P_5 – швидкість вітру в зоні ОП $5 \le V_в \le 10$ м/с;

P_6 – швидкість вітру в зоні ОП $V_в > 10$ м/с;

P_7 – швидкість просування крайки ОП по фронту $V_п < 10$ м/хв;

P_8 – швидкість просування крайки ОП по фронту $10 \le V_п < 15$ м/хв;

P_9 – швидкість просування крайки ОП по фронту $V_п > 15$ м/хв;

P_{10} – пожежа низова середня;

P_{11} – пожежа низова сильна;

P_{12} – пожежа верхова;

P_{13} – об'єкт у небезпеці;

P_{14} – ресурси для гасіння розташовані біля ОП;

P_{15} – ресурси для гасіння розподілені й розташовані на південній межі об'єкта й уздовж перешкод по флангах ОП;

P_{16} – ресурси для гасіння розгорнуті на південній межі об'єкта;

P_{17} – люди і рухоме майно з об'єкта евакуйовані;

P_{18} – пожежа зупинена;

P_{19} – охоплення зони ОП по фронту й флангах проведене;

P_{20} – опорна смуга для пуску зустрічного вогню створена.

Технологічні операції (дії) $\{D_j\}$, $j = \overline{1,10}$ щодо ліквідації пожежі:

D_1 – визначити швидкість вітру в зоні ОП;

D_2 – обчислити швидкість просування крайки ОП по фронту;

D_3 – ідентифікувати тип пожежі;

D_4 – направити ресурси для гасіння ОП;

D_5 – розподілити й спрямувати ресурси для гасіння до південної межі об'єкта й уздовж перешкод по флангах ОП;

D_6 – усі наявні ресурси для гасіння зосередити біля південної межі об'єкта;

D_7 – почати евакуацію людей і рухомого майна з об'єкта;

D_8 – віддати наказ про зупинку пожежі;

D_9 – провести охоплення зони ОП по фронту й флангах;

D_{10} – створити опорну смугу для пуску зустрічного вогню.

Формалізовані правила продукцій з описаного вище фрагмента БЗ виглядають таким чином:

1. ЯКЩО $P_1 = 1$ ТА $P_2 = 1$, ТО D_1, ТОДІ $P_3 = 1$.
2. ЯКЩО $P_3 = 1$ ТА $P_4 = 1$, ТО D_2, ТОДІ $P_7 = 1$.
3. ЯКЩО $P_3 = 1$ ТА $P_5 = 1$, ТО D_2, ТОДІ $P_8 = 1$.
4. ЯКЩО $P_3 = 1$ ТА $P_6 = 1$, ТО D_2, ТОДІ $P_9 = 1$.
5. ЯКЩО $P_7 = 1$, ТО D_3, ТОДІ $P_{10} = 1$.
6. ЯКЩО $P_8 = 1$, ТО D_3, ТОДІ $P_{11} = 1$.
7. ЯКЩО $P_9 = 1$, ТО D_3, ТОДІ $P_{12} = 1$ ТА $P_{13} = 1$.
8. ЯКЩО $P_{10} = 1$, ТО D_4, ТОДІ $P_{14} = 1$.
9. ЯКЩО $P_{11} = 1$, ТО D_5, ТОДІ $P_{15} = 1$.
10. ЯКЩО $P_{12} = 1$, ТО D_6, ТОДІ $P_{16} = 1$.
11. ЯКЩО $P_{13} = 1$, ТО D_7, ТОДІ $P_{17} = 1$.
12. ЯКЩО $P_{14} = 1$, ТО D_8, ТОДІ $P_{18} = 1$.
13. ЯКЩО $P_{15} = 1$, ТО D_9, ТОДІ $P_{19} = 1$.
14. ЯКЩО $P_{19} = 1$, ТО D_8, ТОДІ $P_{18} = 1$.
15. ЯКЩО $P_{13} = 1$ ТА $P_{16} = 1$ ТА $P_{17} = 1$, ТО D_{10}, ТОДІ $P_{20} = 1$.

На рис. 2.1 подано фрагмент НАГ [2], що відображає процес ВНЗ із використанням наведеної вище процедури. Момент досягнення мети (зупинки пожежі) залежить у даному випадку від прогнозу випадкового збурення – швидкості вітру в зоні ОП.

Таким чином, описана процедура дозволяє оцінювати в реальному часі вплив випадкових збурень на процес ВНЗ і відповідно до цього ефективно враховувати неповноту інформації про стан складного об'єкта в екстремальному режимі.

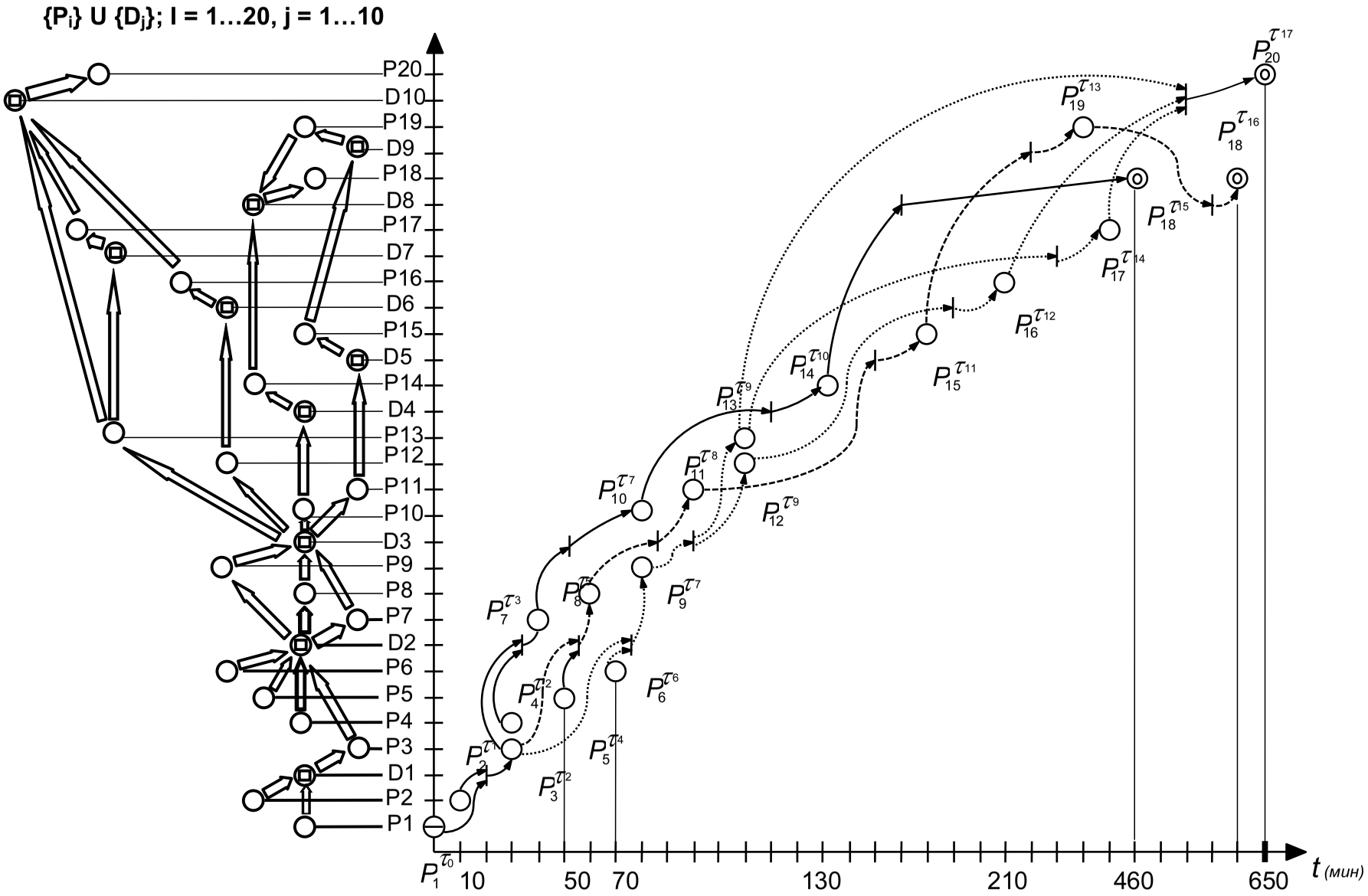


Рис. 2.1. Фрагмент ВНЗ у вигляді НАГ

3. НЕЙРОННІ МЕРЕЖІ

3.1. Структура та математична модель мережі

3.1.1. Нейронні мережі зі зворотним розповсюдженням

Загальновизнаними нейронними мережами, що найширше використовуються, є так звані мережі зі зворотним розповсюдженням (back propagation).

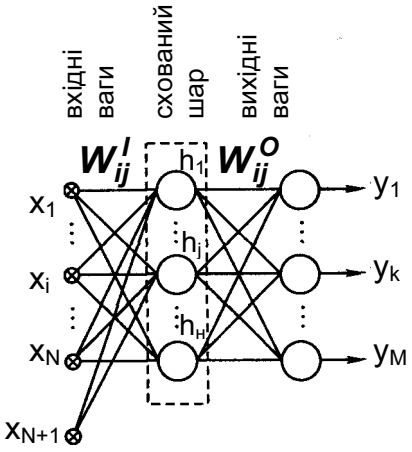


Рис. 3.1. Архітектура нейронної мережі ВР

Ці мережі прогнозують стан фондової біржі, розпізнають почерки, синтезують мову з тексту, керують автомашиною. Далі буде показано, що зворотне розповсюдження скоріше відноситься до алгоритмів навчання, а не до архітектури мережі. Таку мережу правильніше називати мережею з прямою передачею сигналів.

На рис. 3.1 наведено класичну трирівневу архітектуру нейронної мережі. Позначимо: R^d – d-мірний простір.

Вхідній вектор $X = \{x_1, x_2, \ldots, x_N\}$, $x_{N+1} = 1$; вихідний вектор $Y = \{y_1, y_2, \ldots, y_M\}$.

Нейронна мережа виконує функціональне перетворення, яке може бути зображене як $Y = F(X)$, де $X = \{x_i\}, i = \overline{1,N}$; $Y = \{y_k\}$, $k = \overline{1,M}$.

Схований шар насправді може складатися з декількох шарів, проте можна вважати, що достатньо розглянути лише три шари для опису цього типу поведінки. Для нейронної мережі з N вхідними вершинами, J вершинами схованого шару та M вихідними вершинами величини y_k задаються так:

$$y_k = g\left(\sum_{j=1}^{J} W_{jk}^J h_j\right), k = \overline{1,M}, \qquad (3.1)$$

де W_{jk}^J – вихідна вага зв'язку від вершини j схованого шару до вершини k вихідного шару; g – функція (яка буде визначена пізніше), що виконує відображення $R^J \to R'$.

Вихідні сигнали вершин схованого шару h_j, $j = \overline{1,J}$ задаються так:

$$h_j = \sigma\left(\sum_{i=1}^{N} W_{ij}^I x_i + W_j^T\right), j = \overline{1,J}. \qquad (3.2)$$

Тут W_{ij}^I – вхідна вага зв'язку (i, j); W_j^T – величина порога (вага від вузла, що має постійний сигнал, рівний 1, до вузла j); x_i – сигнал на виході i-го вхідного вузла; σ – функція «сигмоїд», що задається так:

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \qquad (3.3)$$

Функція σ в (3.2) називається функцією активації нейронної мережі, іноді її називають функцією запалювання нейронної мережі.

Функція g у рівнянні (3.1) може бути такою ж самою, що й $\sigma(x)$, або іншою. В нашому викладі ми будемо приймати g функцією виду σ, або одиничною функцією, тобто нелінійною. Необхідно, щоб функція активації була нелінійною й мала обмежений вихід, тобто була обмеженою. Графік функції $\sigma(x)$ наведено на рис. 3.2.

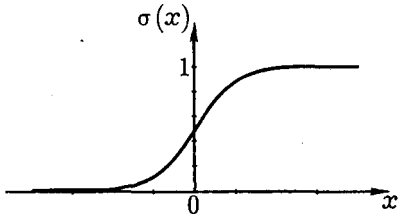


Рис.3.2. Графік функції $\sigma(x)$

3.1.2. Нейронні мережі прямої дії

Функціонування мережі прямої дії визначається двома факторами:

– архітектурою мережі;

– величинами ваг.

Кількість вхідних і вихідних вузлів визначається апріорі та по суті є фіксованою. Кількість схованих вузлів є змінною і може настроюватися (регулюватися) користувачем.

До цього часу це настроювання залишається поки що «мистецтвом», хоча у відповідній літературі були запропоновані різні методи установлення кількості схованих вузлів та вилучення непотрібних.

Після визначення (задання) архітектури мережі власні значення ваг визначають її поведінку.

Кажуть, що мережа *«навчається»,* якщо ваги змінюються так, щоб була досягнута бажана мета. Тут слід мати на увазі, що термін «навчання», запозичений з біології, в мережі означає просте настроювання множини параметрів.

3.2. Градієнтний алгоритм навчання нейронної мережі

Першим алгоритмом, який було розроблено для навчання мережі Back Propagation (BP), є градієнтний метод навчання.

Нехай критерій навчання мережі, що має три шари (один шар прихований), є таким:

$$e(w) = \sum_{i=1}^{M} \left(d_i - y_i(w)\right)^2 \rightarrow min, \qquad (3.4)$$

де d_i – прогнозоване значення i-го виходу нейромережі, $y_i(w)$ – фактичне значення i-го виходу нейромережі BP для вагової матриці $W = \left[W^1; W^0\right]$.

Таким чином, критерій $e(w)$ являє собою середній квадрат помилки апроксимації.

Нехай функції активації для нейронів прихованого шару $h_j = \sigma\left(\sum_{i=1}^{N} W_{ij}^{I} x_i + W_{N+1,j}^{I}\right)$ та нейронів вихідного шару $y_k = \sigma\left(\sum_{j=1}^{J} h_j W_{jk}^{0}\right)$ однакові й становлять функцію «сигмоїд» (3.3).

Для такої функції похідна дорівнює

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)). \tag{3.5}$$

Розглянемо градієнтний алгоритм навчання нейронної мережі.

Алгоритм навчання *H*-мережі.

1. Нехай $W(n)$ – поточне значення матриці ваг. Алгоритм має такий вигляд:

$$W(n+1) = W(n) - \gamma_{n+1} \nabla_w e(W(n)),$$

де γ_n – розмір кроку на n-й ітерації.

2. На кожній ітерації спочатку навчаємо (коригуємо) вхідні ваги:

$$W_{ij}^{I}(n+1) = W_{ij}^{I}(n) - \gamma_{n+1} \nabla_w \frac{\partial e(W)}{\partial W_{ij}^{I}}; \tag{3.6}$$

$$\begin{aligned} \frac{\partial e(W^I)}{\partial W_{ij}^{I}} = -\sum_{k=1}^{M} \{2(d_k - y_k(W^0)) \cdot y_k(W^0) \times \\ \times (1 - y_k(W^0))\} \cdot h_j(W) \cdot (1 - h_j(W)) \cdot x_i . \end{aligned} \tag{3.7}$$

3. Знаходимо (навчаємо) вихідні ваги:

$$\frac{\partial e(W^0)}{\partial W_{jk}^{0}} = -2(d_k - y_k(W^0)) \cdot y_k(W^0) \times (1 - y_k(W^0)) \cdot h_j, \tag{3.8}$$

$$W_{ij}^{0}(n+1) = W_{ij}^{0}(n) - \gamma_{n+1} \frac{\partial e(W)}{\partial W_{ij}^{0}}, \tag{3.9}$$

де $x_i, i = \overline{1, N+1}$ – входи НМ; y_k, $k = \overline{1, M}$ – виходи НМ; h_j, $j = \overline{1, J}$ – виходи прихованого шару.

4. $n := n + 1$ та переходимо на наступну ітерацію.

Зауваження. Так званий алгоритм навчання з пам'яттю має вигляд

$$W(n+1) = W(n) - \lambda(1 - \alpha)\nabla_w e(W(n)) - \alpha \nabla_w e(W(n-1)),$$

де λ – швидкість навчання, $\alpha \in [0;1]$ – параметр забування.

Градієнтний метод є першим запропонованим алгоритмом навчання, він простий в реалізації, але має такі недоліки:

- повільно збігається;
- знаходить лише локальний екстремум.

3.3. Побудова рекурентного виразу для обчислення похідних помилок

Розглянемо градієнтний метод навчання нейронної мережі ВР. Нехай необхідно мінімізувати критерій:

$$E(w) = \frac{1}{2}\sum_p \sum_j \left(y_{pj}^{(N)} - d_{pj}\right)^2, \qquad (3.10)$$

де $y_{pj}^{(N)}$ – реальний вихід j-го нейрона вихідного шару N нейронної мережі при поданні на вхід p-го образу; d_{pj} – бажаний вихід.

Мінімізація здійснюється за методом градієнтного спуску, що означає побудову ваг таким чином:

$$w_{ij}(t) = w_{ij}(t+1) + \nabla w_{ij}, \quad \nabla w_{ij}^{(n)} = -\eta \frac{\partial E}{\partial w_{ij}^{(n)}}, \qquad (3.11)$$

де $w_{ij}^{(n)}$ – ваговий коефіцієнт зв'язку i-го нейрона (n–1)-го шару з j-м нейроном n–го шару; $0 < \eta < 1$ – коефіцієнт швидкості навчання. Тоді

$$\frac{\partial E}{\partial w_{ij}^{(n)}} = \frac{\partial E}{\partial y_j^{(n)}} \cdot \frac{\partial y_j^{(n)}}{\partial s_j^{(n)}} \cdot \frac{\partial s_j^{(n)}}{\partial w_{ij}^{(n)}}, \qquad (3.12)$$

де y_j – вихід j-го нейрона n-го шару; s_j – зважений сумарний вхідний сигнал **j**-го нейрона.

Очевидно, що

$$s_j^{(n)} = \sum_i w_{ij}^{(n)} y_i^{(n-1)}, \qquad (3.13)$$

якщо функція активації j-го нейрона має вигляд

$$y_j = f(s_j), \text{ то } \frac{\partial y_j}{\partial s_j} = f'(s_j).$$

В окремому випадку, якщо $f = \sigma$ – сигмоїд, то

$$\frac{\partial y_j}{\partial s_j} = f'(s_j) = y_j \cdot (1 - y_j).$$

Третій множник $\frac{\partial s_j}{\partial w_{ij}} = y_j^{(n-1)}$.

Що ж стосується першого множника у (3.12) $\frac{\partial E}{\partial y_j^{(n)}}$, то він легко розкладається через виходи нейронів наступного $(n+1)$-го шару таким чином:

$$\frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{ds_k} \cdot \frac{\partial s_k}{\partial y_k} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{ds_k} \cdot w_{jk}^{(n+1)}. \qquad (3.14)$$

Тут підсумовування ведеться по нейронах $(n+1)$-го шару. Введемо нову змінну

$$\delta_j^{(n)} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_k}{ds_k}. \qquad (3.15)$$

Отримаємо рекурентну формулу для обчислення величини $\delta_j^{(n)}$ ***n***-го шару через величини $\delta_j^{(n+1)}$ наступного шару (рис. 3.3.)

$$\delta_j^{(n)} = \sum_{k=1}^{K} \delta_j^{(n+1)} w_{jk}^{(n+1)} \frac{dy_j}{ds_i}. \qquad (3.16)$$

Для вихідного шару $n = N$ маємо

$$\delta_j^{(N)} = \left(y_i^{(N)} - d_i\right)\frac{dy_j}{ds_i}. \qquad (3.17)$$

Тепер можна записати алгоритм градієнтного спуску (3.11) у такому вигляді:

$$\nabla w_{ij}^{(n)} = -\eta \delta_j^{(n)} y_i^{(n-1)}. \qquad (3.18)$$

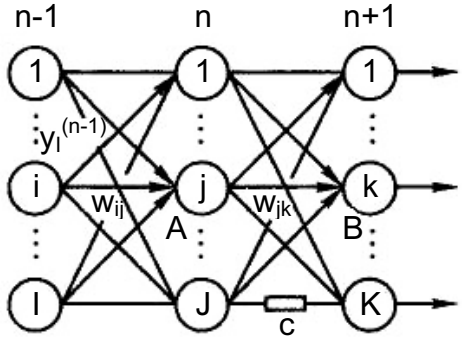


Рис. 3.3. Фрагмент шару ***n*** структури НМ ВР

Іноді для надання процесу корекції ваг деякої інерційності для згладжування різких стрибків, якщо рухатись по поверхні, використовується значення ∇w_{ij} на попередній ітерації $(t-1)$. У цьому випадку величина $\nabla w_{ij}(t)$ буде мати вигляд

$$\nabla w_{ij}^{(n)}(t) = \eta(\mu \Delta w_{ij}^{(n)}(t-1) - (1-\mu)\delta_j^{(n)} y_i^{(n-1)}(t)), \qquad (3.19)$$

де $\mu \in [0,1]$.

Опис алгоритму навчання Back Propagation.

Повний алгоритм навчання НМ за допомогою процедури зворотного поширення складається з таких кроків [23].

Нехай на вхід НМ подано один із можливих образів

$$x = \{x_i\}, i = \overline{1,I}.$$

1. Покладемо $y_i^{(0)} = x_i, i = \overline{1,I}$.

2. Розрахуємо послідовно значення виходів для ***n***-го шару (n = $\overline{1,N}$):

$$s_j^{(n)} = \sum_{i=1}^{I} y_i^{(n-1)} w_{ij}^{(n)}; \qquad (3.20)$$

$$y_j^{(n)} = f(s_j^{(n)}). \qquad (3.21)$$

3. Розрахуємо величини $\delta_i^{(N)}$ для нейронів вихідного шару. Визначимо $\Delta w_{jk}^{(N)}$.

4. Використовуючи рекурентну формулу, розрахуємо $\delta_i^{(n)}$ через $\delta_i^{(n+1)}$ і $\Delta w_{jk}^{(n+1)}$ для всіх попередніх шарів $n = N-1, N-2, \ldots, 1$.

5. Коригуємо ваги в НМ відповідно до процедури:

$$w_{ij}^{(n)}(t) = w_{ij}^{(n)}(t-1) + \Delta w_{ij}^{(n)}(t). \qquad (3.22)$$

На цьому ітерація t закінчується.

6. Розрахуємо $E = E(w(t))$. Якщо $E(w(t)) < \overline{\varepsilon_{зад}^2}$, то СТОП.

Інакше йдемо на крок 1 $(t+1)$-ї ітерації.

Даний алгоритм розрахунку величини $\delta_i^{(n)}$ проілюстровано на рис. 3.4–3.7.

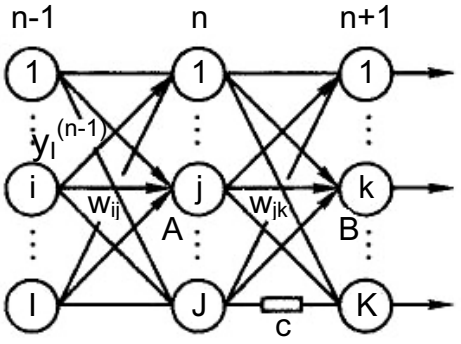


Рис. 3.4. Фрагмент шару n структури НМ ВР

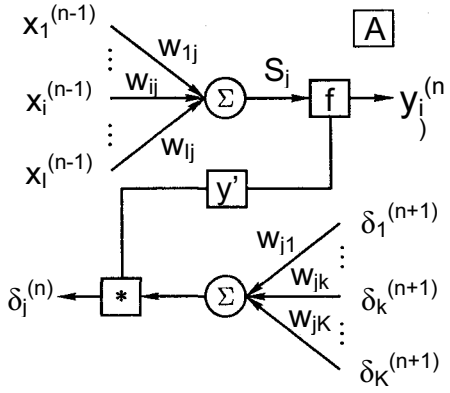


Рис. 3.5. Процедура обчислення $\delta_i^{(n)}$

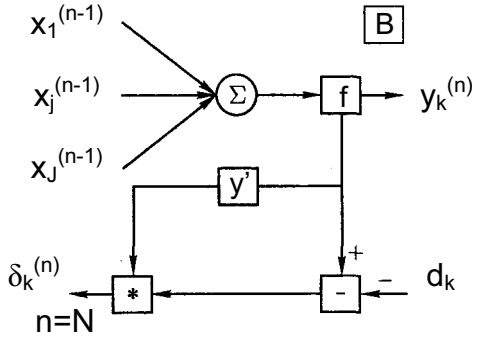


Рис. 3.6. Процедура обчислення $\delta_i^{(n)}$ для останнього шару

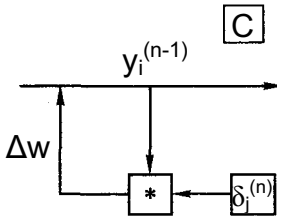


Рис. 3.7. Процедура обчислення ваг Δw_{jk}

3.4. Прискорення збіжності алгоритмів навчання нейронних мереж. Алгоритм спряжених градієнтів

Як показано вище, алгоритм навчання мереж типу «back propagation» – це реалізація класичного методу найшвидшого спуску. Цей алгоритм відносно простий у реалізації й застосуванні, що й пояснює його широке використання в області нейромереж. Однак у нього є два слабких місця: він повільно сходиться та ефективний тільки при пошуку точок локального мінімуму. Тому були розроблені інші, більш ефективні методи навчання, що є альтернативою методу градієнта: метод спряжених градієнтів і метод, заснований на генетичній оптимізації.

Метод спряжених градієнтів.

Метод спряжених градієнтів (СГ) дає поліпшення швидкості збіжності порівняно з методом найшвидшого спуску. Однак, як і метод найшвидшого спуску, він є методом локальної оптимізації.

У нейронних мережах цільова функція (ц.ф.), яку необхідно мінімізувати, – це середня помилка на всій множині навчальних зразків

$$E_{\sum}(W) = \sum_{t=1}^{T}\sum_{k=1}^{M}\left(d_{t_k} - y_{t_k}(W)\right)^2, \qquad (3.23)$$

де $t = \overline{1,T}$ – множина навчальних зразків; M – кількість вихідних вузлів, $\{d_{t_1}, d_{t_2}, \ldots, d_{t_M}\}$ – бажаний вихід для навчального зразка t, а $y_{t_k}(W) = \{y_{t_1}(W), y_{t_2}(W), \ldots, y_{t_M}(W)\}$ – реакція (вихідний сигнал мережі) на зразок t.

Для тришарової мережі з N вхідними вузлами, J схованими вузлами і M вихідними вузлами вектор ваг **W** містить $NJ + MJ$ компонент.

Алгоритм СГ, як і більш загальний алгоритм спряжених напрямків, набув застосування в області оптимізації завдяки широкому класу проблем, для яких він забезпечує збіжність до оптимального рішення за скінчену кількість кроків. Це суттєве поліпшення у порівнянні з методом найшвидшого спуску, що потребує нескінченої кількості ітерацій для пошуку мінімуму функції f.

Спряжені напрямки. Походження назви пов'язане з використанням спряжених векторів. У векторному просторі вимірності D множина векторів $\{P_1, P_2, \ldots, P_D\}$ утворює множину спряжених напрямків щодо матриці A, якщо

$$P_i A P_j = 0, \text{ для } i \neq j, \qquad (3.24)$$

де A – додатно визначена матриця розміром DxD. Вектори, що задовольняють (3.24), називають A-спряженими.

Виникає запитання: яким чином алгоритм СГ досягає збіжності за кінцеву кількість кроків і на яких задачах? Припустимо, що нам необхідно мінімізувати функцію

$$F(W) = (b - AW)^T (b - AW), \qquad (3.25)$$

де b і W – D-вимірні вектори, а матриця A^{DxD} визначена вище. Отже, маємо квадратичну функцію. Припустимо, що шукається ітераційно оптимальний вектор W^*, який мінімізує $E(W)$. Починаємо пошук з початкової точки W_0. Вибираємо ненульовий вектор p_1, який служить напрямком пошуку на наступній ітерації, при цьому не важливо, яким чином були обрані W_0 і p_1. Задамо W_1 як вектор

$$W_1 = W_0 + \alpha p_1, \qquad (3.26)$$

де скаляр α вибирається так, щоб мінімізувати $E(W_0 + \alpha p_1)$. Оптимальний напрямок, у якому необхідно рухатися на наступній ітерації, – це напрямок, де потрібен тільки один крок безпосередньо в точку оп-

тимального розв'язку W^*, і він повинен утворювати A-спряжену пару з вектором p_1.

Оптимальний напрямок – це $W^* - W_1$, тому умова, що $(W^* - W_1)$ є A-спряжений напрямок, еквівалентна твердженню, що повинна виконуватись умова $(W^* - W_1) Ap_1 = 0$.

Звичайно, у цій точці ми не знаємо оптимального розв'язку W^*, у противному разі нам би не треба було ніякого алгоритму. Однак ця умова важлива з такої причини. У D-мірному просторі є рівно D – 1 незалежних векторів, що утворюють A-спряжену пару з вектором p_1.

Таким чином, нам буде потрібна тільки скінчена кількість напрямків, щоб знайти оптимальний розв'язок.

Алгоритм спряжених напрямків систематично конструює множину A-спряжених векторів. Через максимум D кроків алгоритм знайде оптимальний напрямок, і збіжність буде забезпечена.

Тут ми опустили важливе питання визначення задачі одномірної мінімізації по скаляру α. Для задач у формі (3.25) така мінімізація виконується безпосередньо й утворює частину класичного алгоритму СГ, хоча для більш загальних проблем ця задача аж ніяк не тривіальна.

У розглянутій задачі навчання H-мережі не існує в явній формі рівняння (3.25), і, зокрема, ми не маємо явного виразу для матриці A, хоча градієнт помилки ∇E може виконувати цю роль.

Помітимо, що в рівнянні (3.25) AW – множник, пропорційний градієнту функції $E(W)$.

Отже, для квадратичних функцій $E(W)$ метод спряжених градієнтів забезпечує збіжність за скінчену кількість кроків. Проте для функцій загального вигляду скінчена збіжність більше не гарантується.

Необхідно усвідомлювати, що алгоритм СГ, подібно до методу градієнтного спуску, забезпечує знаходження лише локально оптимальних рішень. Проте метод дає значного прискорення збіжності у порівнянні з методом найшвидшого спуску.

Опис алгоритму.

Крок 0. Покласти $K = 0$. Ініціалізувати ваговий вектор W і обчислити градієнт $G = gradE(W)$. Покласти вектор початкового напрямку $p_K = -\frac{G}{\|G\|}$.

Крок 1. Знайти скаляр α^*, що мінімізує $E(W + \alpha p)$, для чого можна використати метод Фібоначчі, або золотого перетину. Покласти

$$W(K+1) = W(K) + \alpha^* p(K)). \qquad (3.27)$$

Крок 2. Якщо $E(W(K+1)) < \varepsilon_{прип}$, де $\varepsilon_{прип}$ – припустима точність досягнення мінімуму, то STOP. Інакше – обчислити новий напрямок

$$G(k+1) = gradE(W(k+1)).$$

Крок 3. Якщо $mod_D(k + 1) = 0$, то новий вектор напрямку

$$P(k+1) = -\frac{G(k+1)}{\|G(k+1)\|},$$

інакше – покласти

$$\beta = \frac{G(K+1)^T G(K+1)}{G(K)^T G(K)} \qquad (3.28)$$

та обчислити новий вектор напрямку

$$P_{(k+1)} = \frac{-G(k+1) + \beta p(k)}{\|-G(k+1) + \beta p(k)\|}.$$

Крок 4. Замінити ***p(k)*** на ***p(k+1)*** та ***G(k)*** на ***G(k+1).*** Перехід на крок 1 наступної ітерації.

3.5. Генетичний алгоритм навчання нейронної мережі

Цей алгоритм є алгоритмом глобальної оптимізації. У ньому використовуються такі механізми [16,22]:

1) схрещування батьківських пар (cross-over), генерація нащадків;
2) мутація (дія випадкових впливів);
3) природний добір кращих (селекція).

Мета навчання – мінімізація середньоквадратичної помилки

$$E(W) = \frac{1}{M}\sum_{k=1}^{M}\left(d_k - y_k(W)\right)^2,$$

де $W = [W_I, W_O]$, $W_I = \left\|w_{ij}^I\right\|$, $W_O = \left\|w_{ij}^O\right\|$.

Задається початкова популяція з ***N*** особин

$$[W_1(0),\ldots,W_i(0),\ldots,W_N(0)].$$

Будь-яку особину можна подати у вигляді числового значення її ваги. Для кожної особини обчислюємо *індекс придатності* (Fitness Index) й оцінюємо якість прогнозування

$$FI(W_i) = C - E(W_i) \to max,$$

де ***C*** – константа.

Схрещування батьківських пар. При виборі батьків використовується ймовірнісний механізм. Позначимо P_i – ймовірність вибору ***i***-го батька:

$$P_i = \frac{FI(W_i(0))}{\sum_{i=1}^{N} FI(W_i(0))}.$$

Потім здійснюється схрещування обраних пар.

Можна застосовувати різні механізми схрещування, наприклад: для першого нащадка беруться непарні компоненти з вектора першого батька, а парні компоненти – з вектора другого батька; для другого нащадка навпаки: парні компоненти – з вектора першого батька, а непарні – з вектора другого батька. Це можна записати таким чином:

$$W_i(0) \otimes W_k(0) \to W_i(1) + W_k(1),$$

$$W_i = \lfloor w_{ij} \rfloor_{j=\overline{1,R}},$$

$$w_{ij}(1) = \begin{cases} w_{ij}(0), \text{ якщо } j = 2m; \\ w_{kj}(0), \text{ якщо } j = 2m - 1; \end{cases}$$

$$w_{kj}(1) = \begin{cases} w_{kj}(0), \text{ якщо } j = 2m; \\ w_{ij}(0), \text{ якщо } j = 2m - 1; \end{cases}$$

$$m = \overline{1, \frac{R}{2}}.$$

Береться $\frac{N}{2}$ батьківських пар і генеруються N нащадків.

Дія мутацій:

$$w'_{ij}(n) = w_{ij}(n) + \xi(n),$$

де $\xi(n) = ae^{-\alpha n}$, $a = random \in [-1; +1]$.

Селекція. Можна використовувати різні механізми селекції:

1. Повна заміна старої популяції на нову.
2. Вибір N кращих із всіх існуючих особин $N_{род} + N_{потомк} = 2N$ за критерієм максимуму FI.

На цьому одна ітерація генетичного алгоритму закінчується.

Описані ітерації повторюємо доти, доки не почне виконуватися одна з таких умов закінчення ітерацій:

а) $\max\limits_{i} FI(w_i(k)) \geq FI_{зад}$, де $FI_{зад}$ – задане значення FI;

б) $k \geq N_{зад}$, де $N_{зад}$ – задана кількість ітерацій (10^3 – 10^4).

Основна перевага генетичного методу – можливість знаходити глобальний мінімум, проте він має такі недоліки:

– потребує значних обчислювальних витрат;

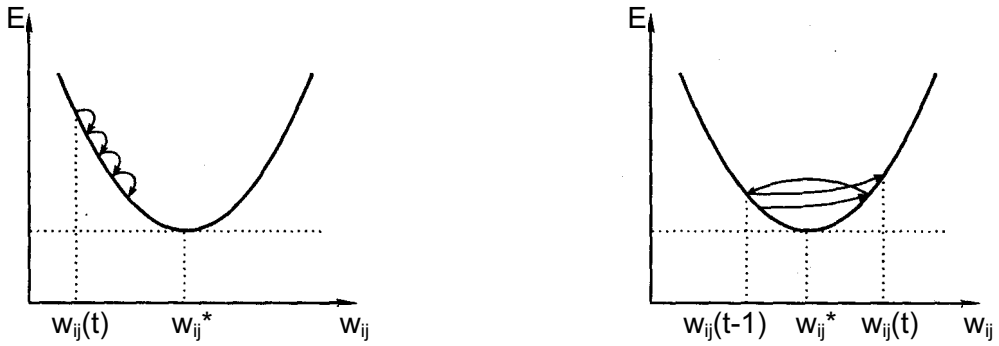
– ряд параметрів визначається експериментально, наприклад: N – розмір популяції; α – показник загасання мутацій.

3.6. Удосконалення градієнтного алгоритму навчання

При реалізації градієнтного алгоритму навчання нейронної мережі ВР може проявитися ряд складностей, властивих градієнтним алгоритмам оптимізації.

1. Якщо ми знаходимося далеко від точки мінімуму функції $E(w_{ij})$, то рухаємося з малим кроком, і процес пошуку може затягуватися. Для його прискорення має сенс збільшити величину кроку $\eta(t)$. Ознакою такої ситуації є сталість знака $\Delta E(t-1) < 0$, $\Delta E(t) < 0$ (рис. 3.8, а).

2. Якщо ж ми знаходимося в околі точки мінімуму w^*_{ij} і величина кроку η велика, то ми перестрибуємо через точку w^*_{ij}, при цьому виникає явище «осциляції». У цьому випадку доцільно поступово зменшувати величину $\eta(t)$. Ознакою такої ситуації є зміна знака ΔE, тобто $\Delta E(t-1)\Delta E(t) < 0$ (рис. 3.8, б).



а б

Рис. 3.8. Ілюстрація роботи градієнтного методу

3.6.1. Градієнтний метод з корекцією кроку навчання (метод відкоту)

Для подолання вищевказаних труднощів, зв'язаних з використанням градієнтного методу, було розроблено градієнтний метод з корекцією кроку навчання. Тут величина кроку $\eta(t)$ на $(t+1)$-й ітерації описується таким рекурентним виразом:

$$\eta(t+1)=\begin{cases}\eta(t)u, \text{ якщо } E(w(t))<E(w(t-1));\\ \eta(t)d \ - \text{ інакше,}\end{cases} \qquad (3.29)$$

де $u > 1$, $0 < d < 1$.

Рекомендується вибирати $ud \approx 1$.

Корекцію кроку можна здійснювати, якщо виконано декілька послідовних кроків, наприклад: $t-2, t-1, t$.

3.6.2. Метод з вибиванням із локальних мінімумів (shock BP)

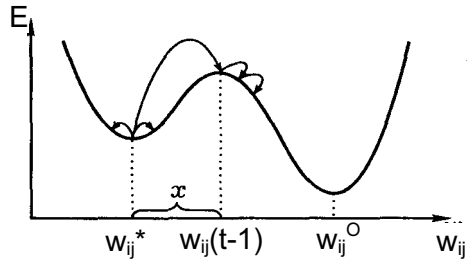


Рис. 3.9. Механізм вибивання з локального мінімуму

Цей метод використовується у випадку багатоекстремальної залежності $E(w)$ при необхідності пошуку глобального мінімуму (рис. 3.9) [23, 32].

У випадку, якщо ми «застрягли» в локальному мінімумі w_{ij}^* і помилка E протягом тривалого часу не змінюється, то має сенс зробити великий крок у випадковому напрямку ξ, щоб «вискочити» з даної положистої западини і потрапити до області притягання іншого мінімуму w_{ij}^*. Тоді $w_{ij}(t+1)=w_{ij}(t)+x\xi$, де ξ рівномірно розподілена в інтервалі [–1; +1].

3.6.3. Метод з векторним кроком навчання (Super SAB)

Основний недолік класичного градієнтного методу полягає в тому, що крок в усіх напрямках однаковий – $\eta(t)$. Він не враховує тієї обста-

вини, що за різними компонентами w_{ij} ми можемо знаходитися на різній відстані від шуканої точки мінімуму (тобто за одними компонентами далеко, а за іншими – близько).

Тому *Almeida* і *da Silva* розробили метод з векторним кроком пошуку, який вони назвали *Super SAB*. У цьому методі пошук відбувається відповідно до виразу

$$w_{ij}(t) = w_{ij}(t-1) - \eta_{ij}(t)\delta_j^{(n)} y_i^{n-1}, \tag{3.30}$$

де

$$\eta_{ij}(t) = \begin{cases} \eta_{ij}(t-1)u, \text{ якщо } \dfrac{\partial E(t)}{\partial w_{ij}} \cdot \dfrac{\partial E(t-1)}{\partial w_{ij}} \geq 0; \\ \eta_{ij}(t-1)d, \text{ якщо } \dfrac{\partial E(t)}{\partial w_{ij}} \cdot \dfrac{\partial E(t-1)}{\partial w_{ij}} < 0, \end{cases} \tag{3.31}$$

$$u > 1, \quad 0 < d < 1.$$

Зміна ваг відбувається відповідно до виразу

$$w_{ij}(t) = \begin{cases} w_{ij}(t-1) - \eta_{ij}(t)\delta_j^{(n)} y_i^{n-1}, \text{ якщо } \dfrac{\partial E(t)}{\partial w_{ij}} \cdot \dfrac{\partial E(t-1)}{\partial w_{ij}} \geq 0; \\ w_{ij}(t-1) \cdot d \text{ – інакше.} \end{cases} \tag{3.32}$$

3.6.4. Автономний градієнтний метод з апроксимацією рельєфу квадратичної функції

Нехай ми знаходимося в точці $w(t)$. Розраховуємо градієнт $\nabla E(w(t))$ і антиградієнт $-\nabla E(w(t))$ і робимо два пробних кроки:

$$w_1 = w(t) + \nabla E(w),$$

$$w_2 = w(t) - \nabla E(w).$$

Обчислюємо $E(w), E(w_1)$ і $E(w_2)$. Далі, припускаючи, що $E(w)$ може бути апроксимована параболою, знаходимо по трьох точках w, w_1 і w_2 точку мінімуму.

Основний недолік методу полягає в тому, що функція $E(w)$ може мати значно складніший вигляд, і тому цю процедуру апроксимації доводиться повторювати багаторазово, що вимагає великих обчислювальних витрат.

3.7. Нейронні мережі, які самоорганізуються. Алгоритм навчання Кохонена

3.7.1. Навчання на основі збігів. Закон навчання Хебба

1949 р. канадський психолог Д. Хебб опублікував книгу «Організація поведінки» (D. Hebb. Organizational Behaviour), у якій він постулював правдоподібний механізм навчання на клітинному рівні в мозку [22, 32].

Основна ідея Хебба полягала в тому, що коли вхідний сигнал нейрона, що надходить через синаптичні зв'язки, викликає спрацьовуван-

ня нейрона, то ефективність такого входу в термінах його здатності сприяти спрацьовуванню нейрона в майбутньому повинна збільшуватися.

Хебб припустив, що зміна ефективності повинна відбуватися саме в синапсі, який подає цей сигнал на вхід нейрона призначення. Пізніші дослідження підтвердили цей здогад Хебба. Хоча останнім часом були відкриті інші механізми біологічного навчання на клітинному рівні, у визнання піонерського характеру заслуг Хебба цей закон навчання було названо на його честь.

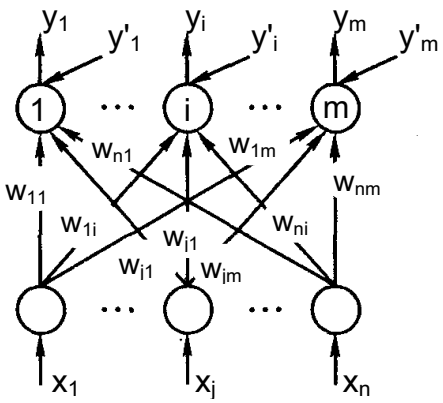


Рис. 3.10. Шар лінійних асоціативних елементів

Закон навчання Хебба належить до класу законів навчання за змаганням.

Лінійні асоціативні елементи. На рис. 3.10 наведено архітектуру нейронної мережі (НМ), яка складається з нейронів, що називаються лінійними асоціаторами. Вхідний вектор у лінійному асоціаторі – це вектор $\boldsymbol{X} = \{\boldsymbol{x_i}\}, \boldsymbol{i} = \overline{\boldsymbol{1,N}}$, що вибирається з простору $\boldsymbol{R^n}$ відповідно до деякого розподілу $\rho(\boldsymbol{X})$.

Вихідний вектор $\boldsymbol{Y}$ обчислюється з вхідного $\boldsymbol{X}$ за такою формулою:

$$\boldsymbol{Y = WX}, \tag{3.33}$$

де $\boldsymbol{W} = \|\boldsymbol{w_{ij}}\|$ – матриця ваг $\boldsymbol{n \times m}$, $\boldsymbol{W = (W_1, W_2, ..., W_m)}$, $\boldsymbol{W_j}$ – стовпці матриці $\boldsymbol{W}$, $\boldsymbol{W_j = (W_{1j}, W_{2j}, ..., W_{nj})^T}$ – вектор ваг.

Позначимо через $\boldsymbol{Y'} = \{\boldsymbol{y_j}\}$ прогнозований вихід. Основна ідея лінійної асоціативної НМ полягає в тому, що мережа повинна навчатися на парах вхід–вихід

$$\boldsymbol{(X_1,Y_1), (X_2,Y_2), ..., (X_L,Y_L).}$$

Коли на вхід НМ подається вхідний сигнал $\boldsymbol{X_k}$, то прогнозований $\boldsymbol{Y'}$ повинен дорівнювати $\boldsymbol{Y_k}$. Якщо на вхід мережі подано вектор $\boldsymbol{X_k} + \varepsilon$ (де ε – достатньо мале), то вихід повинен дорівнювати $\boldsymbol{Y_k} + \varepsilon$ (тобто на виході маємо одержати вектор, близький до $\boldsymbol{Y_k}$.)

Закон навчання Хебба виглядає таким чином:

$$\boldsymbol{w_{ij}^{new} = w_{ij}^{old} + y_{kj} x_{kj}}, \tag{3.34}$$

де $\boldsymbol{x_{kj}}$ – це $\boldsymbol{i}$-та компонента вектора $\boldsymbol{X_k}$; $\boldsymbol{y_{kj}}$ – $\boldsymbol{j}$-та компонента вектора $\boldsymbol{Y_k}$.

У векторному вигляді вираз (3.34) можна записати так:

$$\boldsymbol{W^{new} = W^{old} + X_k Y_k^T = W^{old} + Y_k X_k^T}. \tag{3.35}$$

Щоб реалізувати цей закон у процесі навчання вводять відповідні компоненти $\boldsymbol{Y_k' = [\, y_{kj}\,]}$, які на рис. 3.10. показано стрілками.

Передбачається, що перш ніж почати навчання, усі $w_{ij}^{(0)} = 0$. Тоді в результаті показу навчальної вибірки $(X_1,Y_1), ..., (X_L,Y_L)$ кінцевий стан матриці W задається так:

$$W = Y_1X_1^T + Y_2X_2^T + \ldots + Y_LX_L^T. \quad (3.36)$$

Це рівняння називають формулою суми зовнішнього добутку для W. Її назва виходить з того факту, що $Y_kX_k^T$ – це *зовнішній добуток.*

Переформулювання закону Хебба у вигляді суми зовнішнього добутку дозволяє провести додаткові дослідження можливостей цього закону для забезпечення асоціації пар векторів (X_k,Y_k).

Перший висновок полягає в тому, що коли вектори $\{X_1, X_2, \ldots, X_L\}$ ортогональні й мають одиничну довжину, тобто є ортонормованими, тоді

$$Y_k = WX_k. \quad (3.37)$$

Іншими словами, лінійна асоціативна НМ реалізує бажане перетворення вхід–вихід.

Це наслідок властивості ортонормованості

$$X_i^T X_j = \delta_{ij} = \begin{cases} 0, i \neq j; \\ 1, i = j. \end{cases} \quad (3.38)$$

Тоді

$$WX_k = \sum_{r=1}^{L} Y_r X_r^T X_k = Y_k. \quad (3.39)$$

Але проблема полягає в тому, що умова ортонормованості дуже жорстка (насамперед необхідно, щоб $L \leq n$).

Далі ми обмежені вимогою $\|X_i\| = 1$. Було б набагато корисніше, якби вдалося зняти це обмеження. Ця мета може бути досягнута, але не в лінійній асоціативній НМ. Тут, якщо вектор X_k не є ортонормованим, з'являється помилка відтворення Y_k на виході:

$$WX_k = \sum_{r=1}^{L} Y_r X_r^T X_k = Y_k + \sum_{r \neq k} Y_r X_r^T X_k = Y_k + \eta. \quad (3.40)$$

Бажано домогтися того, щоб η було мінімальним. Для того щоб досягти $\eta = min$ або забезпечити $\eta = 0$, необхідно перейти до асоціативної мережі з нелінійними елементами.

3.7.2. Змагальне навчання

Змагальне навчання використовується в задачах самонавчання, коли немає класифікації вчителя.

Закони навчання, які відносяться до категорії змагальних, мають таку властивість, що виникає змагальний процес між деякими або всіма обробними елементами НМ. Ті елементи, що виявляються переможцями змагання, одержують право змінювати свої ваги. У той час ті, що програли, свої ваги не змінюють (або змінюють за іншим правилом).

Змагальне навчання відоме як «навчання Кохонена» [25]. Навчання Кохонена істотно відрізняється від навчання за Хеббом, або за алгорит-

мом ВР, тим, що в ньому використовується принцип самоорганізації (у противагу принципу контрольованого навчання з учителем) [24].

Змагальний закон навчання має довгу і складну історію. Наприкінці 60-х – початку 70-х рр. Стефан Гроссберг запропонував цілу множину змагальних схем навчання НМ. Іншим дослідником, що займався проблемами змагального навчання, був ван дер Мальсбург. Закон навчання ван дер Мальсбурга був заснований на ідеї, що сума ваг, зв'язаних із входами для різних обробних нейронів єдиного вхідного елемента, повинна залишатися постійною в процесі навчання, тобто якщо одна з ваг (або декілька) збільшується, то інші повинні зменшитися.

Після значних досліджень і вивчення робіт Гроссберга, ван дер Мальсбурга та інших Тойво Кохонен прийшов до висновку, що головна мета змагального навчання має полягати в конструюванні набору векторів, що утворять множину рівноймовірних представників з деякої фіксованої функції щільності розподілу $\rho(X)$ вхідних векторів.

І хоча закони навчання цього типу були отримані незалежно багатьма дослідниками, саме Т. Кохонен був першим, хто звернув увагу на питання про рівноймовірність. Саме завдяки цій ідеї й всесвітньому поширенню книги Т. Кохонена «Самоорганізація й асоціативна пам'ять», його ім'я стало зв'язуватися з даним законом навчання [25].

3.7.3. Закон навчання Кохонена

На рис. 3.11 наведено базову структуру шару Кохонена. Шар складається з N обробних елементів, кожний з яких одержує n вхідних сигналів $x_1, x_2, \ldots, x_n$ *з* нижнього шару, що є прямим передавачем сигналів. Входу x_i і зв'язку (i, j) припишемо вагу w_{ij}.

Кожен обробний елемент шару Кохонена підраховує свою вхідну інтенсивність I_i відповідно до формули

$$I_j = D(W_j, X), \tag{3.41}$$

де $W_j = (w_{1j}, w_{2j}, \ldots, x_{nj})^T$ і $X = (x_1, x_2, \ldots \ldots, x_n)$; $D(W_j, X)$ – деяка міра (метрика) відстані між W_j і X.

Виділимо два вигляди функції $D(W_j, X)$:

1) евклідова відстань:

$$D(W, X) = \|W - X\|;$$

2) сферична дугова відстань:

$$D(W, X) = 1 - W^T X = 1 - \cos\theta, \tag{3.42}$$

де $W^T X$ – скалярний добуток, причому передбачається, що $\|W\| = \|X\| = 1$.

У даному викладі, якщо не зазначено інше, будемо використовувати евклідову відстань $D(X, W)$. При реалізації закону Кохонена як тільки кожен обробний елемент (нейрон) підрахував свою функцію I_j, між ними відбувається змагання, мета якого – знаходження елемента з най-

меншим значенням I_j (тобто $I_{j_{min}}$). Як тільки буде знайдений переможець такого змагання, його вихід z_j покладається рівним одиниці. Вихідні сигнали усіх інших елементів покладаються рівними нулю.

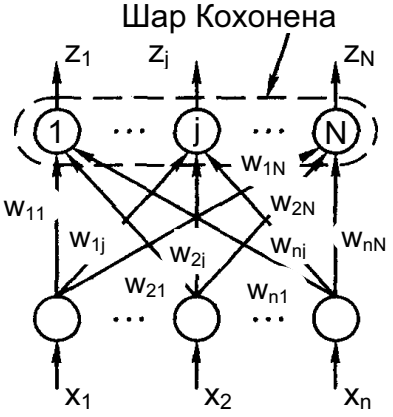


Рис. 3.11. Базова структура шару Кохонена

У цей момент і відбувається навчання по Кохонену.

Навчальні дані для шару Кохонена приблизно складаються з послідовності вхідних векторів $\{X\}$, що витягаються випадково з фіксованою щільністю розподілу ймовірностей $\rho(X)$. Як тільки черговий з векторів X вводиться в мережу, обробні елементи Кохонена починають змагатися між собою, щоб знайти переможця, для якого досягається $\min_j D(X, W_j)$. Тоді для нейрона-переможця j^* встановлюється вихід $z_j^* = 1$, а для всіх інших $z_j = 0, i \neq j^*$.

У цей момент відбувається зміна ваг відповідно до закону навчання Кохонена:

$$W_j^{new} = W_j^{old} + \alpha(X - W_j^{old}) z_j . \qquad (3.43)$$

де $0 < \alpha < 1$.

Даний закон можна переписати у такому вигляді:

$$W_j^{new} = \begin{cases} (1-\alpha) W_j^{old} + \alpha X, \text{ для переможця } j = j^*; \\ W_j^{old}, \forall j \neq j^*. \end{cases} \qquad (3.44)$$

Очевидно, що при такому законі навчання ваговий вектор W_j рухається від W_j^{old} до вхідного вектора X. На початку процесу навчання $\alpha \approx 1$, а потім у міру розвитку процесу навчання зменшується до величини $\alpha = 0{,}1$.

Далі слід зазначити подібність навчання за Кохоненом і статистичного процесу визначення «k-середніх».

K-середні для фіксованого набору векторів $\{X_1, X_2, \ldots, X_L\}$, які обрано випадково з деякої генеральної сукупності з фіксованою щільністю розподілу ймовірностей $\rho(x)$, складають множину k векторів $\{W_1, W_2, \ldots, W_k\}$ таких, що мінімізується функціонал

$$\min_{\{W_i\}} \sum_{i=1}^{L} D^2(X_i, W(X_i)), \qquad (3.45)$$

де $W(X_i)$ – вектор W, найближчий до X_i.

На закінчення необхідно підкреслити, що навчальний закон Кохонена у загальному випадку не генерує множину рівноймовірних вагових векторів, тобто множину таких векторів, що X, який вибрано випадково, відповідно до щільності розподілу ймовірностей ρ буде мати

рівну ймовірність виявитися найближчим до кожного з вагових векторів W_j.

3.7.4. Оцінка щільності розподілу ймовірностей

Як вже відзначалося вище, ми прагнемо домогтися того, щоб отримати вектори W_j , що самі були б приблизно рівноймовірними з точки зору найближчого сусідства стосовно векторів X, які витягаються з R^n з деякою щільністю розподілу ймовірностей $\rho(x)$. Інакше кажучи, для довільного вектора X, витягнутого з R^n з імовірністю $\rho(X)$, бажано, щоб імовірність того, що X виявиться найближчим до W_i, має приблизно дорівнювати $\frac{1}{N}$ для усіх $i = \overline{1,N}$.

Існує кілька підходів для вирішення проблем, що виникають при реалізації базового закону навчання Кохонена [24, 25].

1. Перший підхід називається Radial Sprouting – радіальні паростки. Він є найкращим для евклідової та аналогічних їй метрик (відстаней). Усі вагові вектори $W_j = \lfloor w_{ij} \rfloor$ спочатку вибирають так, що $w_{ij}(0) \approx 0$. Усі вхідні вектори X спочатку помножують на деякий малий позитивний скаляр β. Процес починається з β, близького до нуля. Це забезпечує близькість вхідних векторів X до векторів W_j. У міру розвитку процесу β повільно збільшується, поки не досягне значення β = 1. Як тільки це відбувається, вагові вектори «виштовхуються» зі своїх початкових значень і рухаються за вхідними векторами. Ця схема працює досить добре, але звичайно декілька вагових векторів будуть відставати від процесу й у підсумку виявляться зайвими, що сповільнює процес навчання.

2. Другий підхід (підхід «додавання шуму») полягає в тому, щоб додати рівномірно розподілений шум до векторів даних X, що полегшує ефект досягнення $\rho(X) > 0$ у всій області Ω_x. Рівень (інтенсивність) шуму спочатку вибирають досить великим, щоб вектори шуму були набагато більшими, ніж вектори даних X. Але в міру розвитку процесу навчання рівень шуму поступово знижується. Цей підхід працює правильно, але він виявляється ще повільнішим, ніж підхід «Radial Sprouting». Тому ці підходи вирішують проблему подання законів, які складно уявити, з малою ймовірністю розподілу в деяких областях, але вони не вирішують проблеми рівноймовірного позиціювання векторів W_j.

У цілому базовий закон навчання Кохонена приведе до надлишку при розміщенні векторів W_j у тих областях, де висока щільність розподілу ймовірностей (РЙ) $\rho(X)$, і до нестачі векторів W_j в областях, де щільність $\rho(X)$ мала.

3. Ідея, що була запропонована Дуаном Дез'єном, – це вмонтувати «свідомість» (або пам'ять) у кожен елемент k, щоб здійснити моніторинг (контроль) за історією успішних результатів (перемог) кожного нейроелемента. Якщо обробний елемент Кохонена виграє змагання

істотно частіше, ніж $\frac{1}{N}$ раз (часу), тоді його «свідомість» виключає цей елемент зі змагання на якийсь час, тим самим даючи можливість елементам з перенасиченої області переміщатися до сусідніх ненасичених областей. Такий підхід часто працює дуже добре й у змозі породити досить гарний набір рівноймовірних вагових векторів. Основна ідея механізму «свідомості» – це відстеження частки часу f_j, протягом якого обробний елемент j виграє змагання. Ця величина може бути обчислена локально кожним обробним елементом за формулою

$$f_j(t+1) = f_j(t) + \beta\left(z_j - f_j(t)\right). \qquad (3.46)$$

Відразу ж після того, як змагання закінчено й визначено поточне значення z_j (0 або 1), константа β вибирається малим позитивним числом (типове значення $\beta = 10^{-4} = 0{,}0001$) і обчислюється частка f_j. Далі визначається поточне значення «зсуву» b_j:

$$b_j = \gamma\left(\frac{1}{N} - f_j\right), \qquad (3.47)$$

де γ – позитивна константа (порядку $\gamma \approx 10$).

Далі здійснюється корекція ваг. Однак на відміну від звичайної ситуації, в якій ваги змінюються тільки для одного обробного елемента-переможця з $z_i = 1$, тут проходить окреме змагання для визначення обробного елемента, що має найменше значення величини

$$D(W_j, X) - b_j. \qquad (3.48)$$

Елемент, що виграв, далі коригує свої ваги відповідно до звичайного закону навчання Кохонена.

Роль члена зсуву полягає в наступному. Для елементів j, які часто виграють, величини $f_j > \frac{1}{N}$ і $b_j < 0$, тому для них величина $D(W_j, X) - b_j$ зростає порівняно з $D(W_j, X)$, тоді як для елементів, які рідко виграють, $f_j << \frac{1}{N}$, $b_j > 0$ і $D(W_j, X) - b_j$ зменшуються, що підвищує їхні шанси на перемогу.

3.7.5. Розвиток алгоритму Кохонена

1982 р. Т. Кохонен запропонував ввести до базового правила змагального навчання інформацію про розташування нейронів у вихідному шарі. Для цього нейрони вихідного шару впорядковуються, утворюючи одновимірні або двовимірні грати. Розташування нейронів у таких гратах маркується векторним індексом $i = (i_1, i_2)$. Таке упорядкування природно встановлює відстань між нейронами $|i - j|$.

Модифіковане правило змагального навчання Кохонена враховує відстань нейронів від нейрона-переможця [24, 32]:

$$W_j(t+1) = W_j(t) + \alpha(t)\left(X - W_j\right)\Lambda\left(d\left(i, j^*\right)\right), \qquad (3.49)$$

де Λ – функція сусідства, $\Lambda(d(i, j^*))$ дорівнює одиниці для нейрона-переможця з індексом j^* і поступово убуває в міру збільшення відстані d, наприклад, за законом

$$\Lambda(d) = e^{-d^2/R^2}.$$

Як темп навчання α, так і радіус взаємодії R поступово зменшуються в процесі навчання, так що на кінцевій стадії навчання ми повертаємося до базового закону адаптації вагів тільки нейронів-переможців: $R(t) = R_0 e^{-kt}$.

На відміну від газоподібної динаміки при індивідуальному підстроюванні нейронів навчання по Кохонену нагадує натягування еластичної сітки прототипів на масив даних з навчальної вибірки. У процесі навчання еластичність сітки поступово зменшується.

У результаті одержуємо не тільки квантування входів, але й упорядковуємо вхідну інформацію у вигляді одновимірної або двовимірної топографічної карти Кохонена. На цій сітці кожен багатомірний вектор має свою координату, причому чим ближче координати двох векторів на карті, тим ближче вони й у вихідному просторі.

Така топографічна карта дає наочне уявлення про структуру даних у багатовимірному вхідному просторі, геометрію якого ми не в змозі уявити собі інакше. Візуалізація багатовимірної інформації є головним застосуванням карт Кохонена [24, 25].

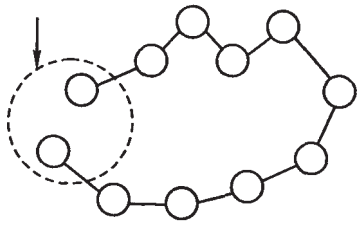


Рис. 3.12. Приклад порушення неперервності відображення

Зауважимо, що відповідно до загального життєвого принципу «безкоштовних обідів не буває» топографічні карти зберігають відношення близькості лише локально, тобто близькі на карті області є близькими й у вихідному просторі, але не навпаки (рис. 3.12). У загальному випадку не існує відображення, що понижує розмірність і зберігає при цьому відношення близькості глобально.

На цьому рисунку стрілкою показано область порушення неперервності відображення: близькі на площині точки відображаються на протилежні кінці карти.

Зручним інструментом візуалізації є розфарбування топографічних карт аналогічно тому, як це робиться на звичайних географічних картах. Кожна ознака породжує своє розфарбування карти – за величиною середнього значення цієї ознаки у даних, що потрапили у певний осередок.

Зібравши докупи карти всіх ознак, що нас цікавлять, одержимо топографічний атлас, який дає інтегральне уявлення про структуру багатовимірних даних. Самонавчальні мережі Кохонена широко використовуються для попередньої обробки даних при розпізнаванні образів у просторі дуже великої розмірності. У цьому випадку, щоб процедура

була ефективною, потрібно спочатку зкомпресувати вхідну інформацію одним із способів:

– пониженням розмірності, виділивши значущі ознаки;
– квантуванням даних.

Розглянемо **звичайний алгоритм навчання Кохонена**:

$$W_{ij}(t+1) = W_{ij}(t) + \alpha\left(y_i^{i-1} - W_{ij}(t)\right)z_j, \qquad (3.50)$$

де y_i^{n-1} – вихід i-го нейрона попереднього шару;

$$z_j = \begin{cases} 1, \text{ якщо } D(W_j, y^{(n-1)}) = \min\limits_i D(W_i, y^{(n-1)}); \\ 0 - \text{інакше.} \end{cases}$$

При використанні навчання за алгоритмом Кохонена існує практика нормалізації вхідних образів:

$$X_i^n = \frac{X_i}{\|X_i\|}, \qquad (3.51)$$

а також ініціалізації вагових коефіцієнтів w_{ij}. Ініціалізація вагових коефіцієнтів випадковими значеннями може призвести до того, що різні класи, яким відповідають щільно розташовані вхідні образи або зіллються, або, навпаки, роздрібняться на додаткові підкласи у випадку близьких образів того самого класу.

Для запобігання цього явища використовується метод опуклої комбінації. Його суть полягає в тому, що вхідні нормалізовані образи піддаються перетворенню вигляду

$$X_i = (1 - \alpha(t))X_i + \alpha(t)\frac{1}{\sqrt{n}}, \qquad (3.52)$$

де $\alpha(t)$ – коефіцієнт, що змінюється у процесі навчання від одиниці до нуля.

У результаті цього спочатку на входи мережі подаються майже однакові образи, а з часом вони усе більше збігаються до вихідних.

Вагові коефіцієнти на кроці ініціалізації встановлюють такими, що дорівнюють $w_0 = \frac{1}{\sqrt{n}}$, де n – розмірність вектора вагів для нейронів ініціалізуючого шару.

На основі розглянутого методу будуються H-мережі особливого типу – так звані карти ознак, що самоорганізуються (self-organizing feature maps). Для них після вибору з шару n нейрона j з мінімальною відстанню $D_j = D(W_j, Y^{(n-1)})$ навчається за формулою (3.50) не тільки цей нейрон, але і його найближчі сусіди, що займать осередок радіусом R. Величина R на перших ітераціях дуже велика, так що навчаються спочатку всі нейрони, але з часом R зменшується до 0. Таким чином, чим ближче закінчення навчання, тим точніше визначається група нейронів, що відповідають кожному класу образів.

Експериментальні дослідження алгоритму самонавчання Кохонена. Розглянемо результати використання алгоритму Кохонена в задачі розпізнавання літер.

Після навчання на еталонних і зашумлених зразках (при 5 %-му рівні шуму) було проведено експерименти по розпізнаванню:

для 5 літер	АПРОЛ;
для 10 літер	ФЫВАПРОЛДЗ;
для 25 літер	ЙЦУКЕНГШЩЗХЬФЫВАПРОЛДЯЧСМ;
для 32 літер	ЙЦУКЕНГШЩЗХЬФЫВАПРОЛДЖЗЯЧСМИТЬБЮ.

При цьому рівень шуму змінювався від 0 до 25 %. У табл. 3.1–3.8 наведено кількість правильно розпізнаних літер А П Р О Л у відсотках для різних вихідних алфавітів, а також середній відсоток правильного розпізнавання (позначено як Ср).

Таблиця 3.1

Кількість правильно розпізнаних зображень для алфавіту з п’яти літер при навчанні за еталонними зразками

Шум,%	А	П	Р	О	Л	Ср.,%
0	100	100	100	100	100	100
5	100	100	100	100	100	100
10	100	100	100	100	100	100
15	100	100	100	100	100	100
20	100	100	100	100	100	100
25	100	100	100	100	100	100

Таблиця 3.2

Кількість правильно розпізнаних зображень для алфавіту з п’яти літер при навчанні за зашумленими зразками

Шум,%	А	П	Р	О	Л	Ср.,%
0	100	100	100	100	100	100
5	100	100	100	100	100	100
10	100	95	100	100	100	99
15	100	90	95	95	100	96
20	100	85	95	90	95	93
25	100	85	85	85	90	89

Таблиця 3.3

Кількість правильно розпізнаних зображень для алфавіту з 10 літер при навчанні за еталонними зразками

Шум,%	А	П	Р	о	л	Ср.,%
0	100	100	100	100	100	100
5	100	100	100	100	100	100
10	100	100	100	100	100	100
15	100	100	100	100	100	100
20	100	100	100	100	100	100
25	100	100	100	100	100	100

Таблиця 3.4

Кількість правильно розпізнаних зображень для алфавіту з 10 літер при навчанні за зашумленими зразками

Шум, %	А	П	Р	О	Л	Ср.,%
0	100	100	100	100	100	100
5	100	100	100	100	100	100
10	100	90	90	95	100	95
15	100	90	85	90	95	92
20	100	85	85	85	90	89
25	95	70	75	85	85	82

Таблиця 3.5

Кількість правильно розпізнаних зображень для алфавіту з 25 літер при навчанні за еталонними зразками

Шум,%	А	П	Р	0	Л	Ср.,%
0	100	100	100	100	100	100
5	100	100	100	100	100	100
10	100	100	100	100	100	100
15	100	95	100	100	100	99
20	100	90	95	95	100	96
25	100	90	95	85	95	93

Таблиця 3.6

Кількість правильно розпізнаних зображень для алфавіту з 25 літер при навчанні за зашумленими зразками

Шум,%	А	П	Р	О	Л	Ср.,%
0	100	100	100	100	100	100
5	100	90	100	100	100	98
10	100	65	90	85	100	88
15	100	60	80	80	90	82
20	100	45	80	55	80	72
25	95	35	70	ЗО	75	61

Таблиця 3.7

Кількість правильно розпізнаних зображень для алфавіту з 32 літер при навчанні за еталонними зразками

Шум, %	А	П	Р	0	Л	Ср.,%
0	100	100	100	100	100	100
5	100	100	100	100	100	100
10	100	100	100	100	100	100
15	100	95	100	100	100	99
20	100	90	95	90	100	95
25	100	85	95	85	95	92

Таблиця 3.8

Кількість правильно розпізнаних зображень для алфавіту з 32 літер при навчанні за зашумленими зразками

Шум,%	А	П	Р	0	Л	Ср.,%
0	100	100	100	100	100	100
5	100	85	90	95	100	94
10	100	60	85	90	100	87
15	100	45	65	90	90	78
20	100	40	60	80	80	72
25	90	20	40	70	75	59

3.8. Застосування нейронних мереж у задачах прогнозування в макроекономіці

Одним з найпоширеніших застосувань нейронної мережі ВР є прогнозування макроекономічних, а також інших часових випадкових процесів.

Особливості прогнозування процесів в економіці перехідного періоду полягають у такому:

1. Процеси, що прогнозуються, відносяться до класу нестаціонарних процесів, параметри яких змінюються з часом. Це особливо характерно для країн з перехідною економікою.

2. Макроекономічні процеси держави тісно пов'язані один з одним, проте характер цієї залежності між прогнозованим наслідком та вхідними умовами складний, і вигляд цієї залежності, як правило, невідомий особі, що приймає рішення.

3. Вихідна інформація про ці процеси є неповною, як правило, недостовірною й обмеженою в часі (так звані короткі виборки).

За цих обставин неможливо використати для прогнозування класичні методи статистичного аналізу, зокрема, кореляційний та регресивний, для яких необхідно, щоб процеси були стаціонарними і був відомий вигляд залежності між вхідними та вихідними процесами.

Тому для задач прогнозування нестаціонарних часових процесів з невідомою структурою прогнозуючої моделі перспективним є використання нєйронних мереж, які не потребують знання вигляду математичної залежності й дуже просто настроюються шляхом навчання та корекції вагів.

Вагові коефіцієнти мережі.

Відомо, що вагові коефіцієнти відіграють роль пам'яті ***H***-мережі. Зрозуміло, обсяг пам'яті прямо пропорційний кількості цих коефіцієнтів, а оскільки ми маємо справу з мережами ВР фіксованої архітектури, то можна сказати, що при фіксованій кількості входів і виходів мережі цей обсяг залежить тільки від кількості внутрішніх шарів і розмірності прихованого шару.

На жаль, не існує чіткого правила задання значень цим двом параметрам, їхній вибір залишається мистецтвом балансування між достатністю й надмірністю.

Проте вибір необхідного обсягу пам'яті варто робити з огляду на розмір вікна, кількість входів і виходів мережі, а також діапазон значень прогнозованої величини, оскільки чим більше діапазон значень, тим більше пам'яті потрібно для чіткого запам'ятовування цих коливань. Безумовно, добір параметрів обсягу пам'яті краще робити експериментально.

Розглянемо підходи до навчання ***H***-мереж, що використовують технологію ковзного вікна [15, 16].

Перший підхід полягає в мінімізації сумарної помилки на вікні навчання, що малоефективно, тому що при такому алгоритмі ми дійсно одержимо не мережу-оракул, а мережу-апроксиматор.

Справа полягає у такому:

– у вікно можуть попадати нерепрезентативні пари-приклади з іншого закону взаємовпливу;

– закон взаємовпливу в межах вікна може різко змінюватися, отже, будуть існувати пари, на яких неможливо мінімізувати помилку окремо, тобто неможливо підібрати мережу, добре навчену під кожну з таких пар.

Таким чином, оскільки ***H***-мережа – це все-таки універсальний апроксиматор, мінімізація помилки на всіх зразках може привести до перенавчання мережі. Ми одержимо мережу-функцію, що апроксимує тренувальну послідовність, у тому числі й нерепрезентативні точки. Мережа не відкриє закон взаємовпливу і, як наслідок, не зможе зробити якісний прогноз. Область застосування такого алгоритму навчання звужена до прогнозування параметрів з невеликим діапазоном значень. Однак якісно спрогнозувати показник, що має відношення до економіки перехідного періоду, з таким підходом дуже проблематично.

Другий підхід полягає в мінімізації помилки на кожній парі-прикладі вікна у порядку їхнього проходження. Проте, рухаючись ітеративно по вікну, мережа буде забувати інформацію, отриману від перших точок вікна на той час, коли дійде черга навчатися на останніх.

Третій підхід полягає в мінімізації помилки на кожній парі-прикладі вікна, але з тасуванням цих пар по визначеному закону, що змінюється залежно від прогресу навчання. Наведені нижче результати експериментів свідчать про високу ефективність такого підходу порівняно з іншими підходами. Можна навіть говорити про нього як про найбільш адекватний. При такому алгоритмі ми автоматично застраховані від ефекту перенавчання ***H***-мережі, тому що вона просто не стане навчатися на нерепрезентативних парах-прикладах.

Прогнозувати *майбутнє,* виходячи з закону, укладеного в парах-прикладах, а не *наступне значення функції* з упорядкованого ряду попередніх її значень – ось мета навчання ***H***-мережі.

Опис алгоритму навчання *H*-мережі.

Розглянемо алгоритм навчання нейромережі ВР для задач прогнозування в макроекономіці, розроблений у роботі [15].

Насамперед необхідно відзначити, що даний алгоритм складається з двох модулів: тасування пар-прикладів і генетичного алгоритму глобальної оптимізації.

Алгоритм тасування пар-прикладів.

Задання алгоритму полягає в тому, щоб примусити ***H***-мережу запам'ятати всі приклади і перетворитися в модель, що являє собою «чорну шухляду», яка вміє на парах-прикладах правильно зіставляти вхід і потрібний вихід. При цьому важливо не перенавчити мережу. Висловлюючись образно, мережа повинна увібрати в себе тільки основний зміст, закладений у парах-прикладах, і пропустити непотрібні подробиці, що засмічують пам'ять.

Тут можна провести паралель зі здатністю людини використовувати аналогію при прийнятті рішень у ситуаціях, не відомих їй раніше.

Насамперед варто сказати, що цей алгоритм відноситься тільки до пар-прикладів, до кожної з яких у свою чергу застосовується алгоритм оптимізації, обраний користувачем.

Схема алгоритму тасування точок на вікні:

1. До кожної пари-прикладу застосовується алгоритм оптимізації, у результаті якого одержуємо ваговий вектор, коли помилка на даній парі-прикладі менше або дорівнює заданій $\xi(\boldsymbol{W})$. Таким чином, перший раз пройшовши по вікну послідовно по кожній парі-прикладу, одержуємо ваговий вектор ***W***.

2. Далі, використовуючи отриманий ***W***, ще раз проходимо по вікну і складаємо масив помилок, що відповідає парам-прикладам, у результаті одержуючи деяку схему пріоритетів за значеннями помилок.

3. Тепер знову проходимо по парах-прикладах оптимізуючим алгоритмом, але вже не послідовно, а відповідно до значень масиву помилок, а саме: починаємо з пари, якій відповідає найбільша помилка, і закінчуємо парою з найменшою помилкою. При цьому пари, на яких на даній ітерації досягнута задана точність, у навчанні не приймають участь. Після цього кроку одержуємо новий ваговий вектор ***W*** і переходимо до п. 2.

4. Навчання закінчується після досягнення заданої точності на кожній парі-прикладі або якщо перевищена припустима кількість ітерацій алгоритму тасування, що задана користувачем.

Модифікація генетичного алгоритму.

У пропонованому генетичному алгоритмі [15, 16], як і в інших алгоритмах цього класу, існує стандартний набір параметрів, що варіюються: розмір популяції, рівень мутації, алгоритм зміни рівня мутацій.

Модифікація даного алгоритму полягає в застосуванні спрощеної схеми зміни рівня мутацій. Користувач задає початковий і кінцевий рівні мутацій в умовних одиницях та кількість ітерацій алгоритму, при яких рівень мутацій зменшиться з початкового значення до кінцевого. При цьому зменшення рівня мутацій відбувається за лінійним законом вигляду ***y = kx + b***.

Результати експериментів.

Як підтвердження правильності вибору нами третього підходу наведемо зведені таблиці результатів експериментів для різних алгоритмів навчання [15].

Вихідні дані, прогнозоване значення, лагові періоди, тип і параметри ***H***-мережі були ідентичні в цих експериментах. Розходження полягало винятково в тому, що створена авторами програма Black Oracle Pro^{T}M (BPO) використовує при навчанні третій підхід, а альтернативні програми – перший підхід.

Вихідні дані. П'ять показників економіки України (агрегат М0, агрегат М2, індекс оптових цін [ІОЦ], індекс споживчих цін [ІСЦ], кредити, вкладені в економіку [КВВЕ]) подані у вигляді часових рядів. Дані охоплюють період з 01.01.95 по 01.12.97, виміри винонувались щомісяця. За прогнозовану величину був узятий показник ІСЦ (табл. 3.9).

Таблиця 3.9

Фрагмент вихідних даних

№ п/п	ДАТА	М0	М2	ІОЦ	ІСЦ	КВВЕ
1	01.01.95	4,70	2,00	29,20	21,20	2,10
2	01.02.95	15,40	13,20	11,40	18,10	13,40
3	01.03.95	18,70	8,00	9,30	11,40	10,10

*Лагова модель **H**-мережі:* вхід (М0[-7], М2[-7], ІОЦ[0], КВВЕ[-7], ІСЦ[0]), вихід (ІСЦ[+1]).

Схема експерименту: використовується ковзне вікно розміром 12 пар-прикладів, що становить 12 місяців, потрібно прогнозувати 13-ту точку. Здвигаючи вікно з одиничним кроком, спрогнозуємо 11 точок. Як критерій оптимізації навчального алгоритму використовується точність прогнозу – СКП (середньоквадратична похибка).

Параметри мережі: число входів – 5, число виходів – 1, розмірність внутрішнього шару – 10, точність навчання пари-приклада – 10^{-5}.

Параметри генетичного алгоритму пакета BOP: розмір популяції – 40, початковий рівень мутацій – ***y*** = 320, кінцевий рівень мутацій – ***y*** = 20, передбачуване число ітерацій на вікні – 1000, закон зміни рівня мутацій – лінійний.

Зведені результати експериментів по навчанню нейромережі при різних алгоритмах навчання наведено в табл. 3.10, а результати прогнозування за цими алгоритмами навчання – в табл. 3.11.

Таблиця 3.10

Результати навчання ***H***-мережі

Алгоритм навчання ***H***-мережі	Значення СКП повіконно											
	№ 1	№ 2	№ 3	№ 4	№ 5	№ 6	№ 7	№ 8	№ 9	№ 10	№ 11	№ 12
BP	0,01004	0,0005	0,00017	0,00011	0,0036	0,0001	0,00941	0,00404	0,00046	0,00086	0,00632	0,0356
BP evo	0,01758	0,01688	0,01578	0,01212	0,0035	0,00127	0,00725	0,00562	0,00471	0,00404	0,00579	0,09453
BP fall back	0,08545	0,05997	0,07233	0,07759	0,08534	0,07547	0,0758	0,07996	0,07768	0,07525	0,08004	0,84486
BP fall back evo	0,09106	0,06115	0,04839	0,05092	0,05001	0,03832	0,03548	0,03315	0,03654	0,0338	0,03769	0,51651
BP vector step	0	0	0	0	0	0	0	0	0	0	0,00001	0,00001
BP vector step evo	0,00206	0,00334	0,00287	0,00255	0,00179	0,00072	0,00262	0,00389	0,00301	0,00157	0,00156	0,02588
BP shock	0,01259	0,00565	0,00054	0,00943	0,00171	0,00112	0,00875	0,01279	0,00413	0,01162	0,00853	0,07684
BP shock evo	0,08801	0,04254	0,03813	0,03491	0,03113	0,02847	0,02046	0,02046	0,02046	0,02046	0,02046	0,36546
HJ evo	0,06618	0,04055	0,03973	0,04308	0,04422	0,03408	0,02023	0,027	0,03022	0,02356	0,02488	0,39373
Heuristic HJ	0,07061	0,10169	0,10971	0,11653	0,11831	0,11226	0,11599	0,11461	0,18192	0,17575	0,18347	1,40083
Heuristic HJ evo	0,00728	0,01384	0,01852	0,01079	0,0157	0,00844	0,00749	0,0078	0,01978	0,02143	0,0128	0,14386
BO PRO BP	0,09296	0,06322	0,056	0,05212	0,05097	0,04258	0,02884	0,02731	0,02716	0,01811	0,00893	0,46819
BO PRO GEN	0,08877	0,05318	0,04984	0,04853	0,0489	0,03028	0,02535	0,02767	0,02436	0,02144	0,02295	0,44127
BO PRO GEN+BP	0,0793	0,0508	0,0478	0,0461	0,0446	0,0295	0,0217	0,02507	0,02551	0,02428	0,02042	0,41508

Таблиця 3.11

Результати прогнозів ***H***-мережі

Алгоритм навчання ***H***-мережі	Номер прогнозованої точки										
	13	14	15	16	17	18	19	20	21	22	23
	Реальне значення ІСЦ										
	2	1,5	1,2	0,9	2,2	1,2	0,1	0,8	0,8	0,1	0,1
	Прогнозні значення ІСЦ										
BP	-2,61693	16,7802	4,06173	-0,29532	-1,4108	0,73349	0,83411	2,9649	1,02889	2,86605	1,61711
BP evo	-2,5154	18,484	1,4729	0,30536	0,89443	-0,23587	1,23837	1,49985	0,95191	0,90922	4,40915
BP fall back	1,60667	5,4299	3,66196	3,49016	3,4931	3,53568	3,42473	1,69144	2,88601	3,08078	3,26891
BP fall back evo	1,88306	3,51883	3,45194	3,24742	3,18056	3,55969	3,49103	2,56382	3,10869	3,26143	2,96084
BP vector step	-0,38645	6,9876	0,14191	-0,05191	1,04855	0,81644	1,75377	13,4115	0,89207	0,43113	2,33873
BP vector step evo	-2,59518	20,5864	7,77759	7,07751	1,82606	-0,15846	1,09559	1,02163	0,88354	0,49311	2,27434
BP shock	-1,57723	21,2383	13,4517	-1,34041	1,63206	0,08989	1,52497	0,52489	0,5441	0,68132	1,4407
HJ evo	-1,04974	3,20592	1,70183	1,13786	0,9054	1,09172	1,32837	-0,643	1,01506	2,12583	1,74362
Ctrld HJ	23,8441	4,06094	3,39974	1,46598	1,85068	2,41256	3,93228	23,85	0,80862	2,55357	1,63993
Ctrld HJ evo	-0,18824	23,85	-0,40545	3,33762	2,79692	4,79724	-0,08866	9,39836	0,77528	1,19131	1,31284
BO PRO BP	**0,20638**	**3,63469**	**0,20638**	**1,65343**	**0,84505**	**1,28644**	**1,71371**	**0,22254**	**0,47426**	**0,61324**	**1,49207**
BO PRO GEN	**0,48119**	**1,83151**	**1,26425**	**0,73327**	**0,76671**	**1,15991**	**1,3411**	**0,35309**	**0,48425**	**0,74165**	**0,92018**
BO PRO GEN + BP	**0,4431**	**1,55333**	**1,13124**	**0,63042**	**0,59576**	**1,11238**	**1,40165**	**0,38367**	**0,52715**	**0,5687**	**0,76824**

Таблиця 3.12

Результати порівняння реального і прогнозного значень ІСЦ

Алгоритм навчання ***H***-мережі	Номер дельти прогнозу											
	Delta 13	Delta 14	Delta 15	Delta 16	Delta 17	Delta 18	Delta 19	Delta 20	Delta 21	Delta 22	Delta 23	Total Delta
	Модулі різниці реального і прогнозного значень											
BP	4,61693	15,2802	2,86173	1,19532	3,6108	0,46651	0,73411	2,1649	0,22889	2,76605	1,51711	35,4425
BP evo	4,5154	16,984	0,2729	0,59464	1,30557	1,43587	1,13937	0,69985	0,15191	0,80922	4,30915	32,2168
BP fall back	0,39333	3,9299	2,46196	2,59016	1,2931	2,33568	3,32473	0,89144	2,08601	2,98078	3,16891	25,456
BP fall back evo	0,11694	2,01833	2,25194	2,34742	0,98056	2,35969	3,30103	1,7682	2,30869	3,16143	2,86084	23,5606
BP vector step	2,38465	5,4876	1,05809	0,95191	1,15145	0,38356	1,65377	12,6115	0,09207	0,33113	2,23873	28,3462
BP vector step evo	4,59518	19,0864	6,57759	6,17751	0,37394	1,35846	0,99559	0,22163	0,08354	0,39311	2,17434	42,0372
BP shock	3,57723	19,7383	12,2517	1,24041	0,56794	1,11011	1,42497	0,27512	0,2559	0,58132	1,3407	42,3637
HJ evo	3,04974	1,70592	0,50183	0,23786	1,2946	0,10828	1,22837	1,44299	0,21506	2,02583	1,64362	13,4541
Ctrld HJ	21,8441	2,56094	2,19974	0,56598	0,34932	1,21256	3,83228	23,05	0,00862	2,45357	1,53993	59,6170
Ctrld HJ evo	2,18824	22,35	1,60545	2,43762	0,59692	3,59724	0,18866	8,59836	0,02472	1,09131	1,21284	43,8913
BO PRO BP	**1,79362**	**2,13469**	**0,99362**	**0,75343**	**1,35495**	**0,08644**	**1,61371**	**0,57746**	**0,32574**	**0,51324**	**1,39207**	**11,5389**
BO PRO GEN	**1,51881**	**0,33151**	**0,06425**	**0,16673**	**1,43329**	**0,04009**	**1,2411**	**0,44681**	**0,31575**	**0,64165**	**0,82018**	**7,02026**
BO PRO GEN+BP	**1,5569**	**0,05333**	**0,06876**	**0,26958**	**1,60424**	**0,08762**	**1,30165**	**0,41633**	**0,27285**	**0,4687**	**0,66824**	**6,76819**

Умовні позначки:

BP – алгоритм Back Propagation;

evo – еволюційний;

fall back – з методом відкоту;

Heuristics – з евристикою;

shock – шоковий метод;

HJ – метод Хука–Дживса;

BO PRO – позначення стосується результатів, отриманих за допомогою BOPro™;

GEN – генетичний алгоритм.

В усіх експериментах використовувався механізм ковзного вікна – розмір вікна навчання – 12 точок, 13-та точка була прогнозна і по ній оцінювалась якість прогнозу. Далі вікно здвигалося на одну точку вперед, і процес навчання повторювався.

Висновок.

Проаналізувавши результати, можна зробити кілька важливих висновків.

При використанні підходу з мінімізацією сумарної функції помилки Н-мережа, як правило, перенавчається і, незважаючи на відмінну якість навчання, прогнозовані значення далекі від реальних. Іншими словами, якісне навчання ще не означає успіх при прогнозуванні (табл. 3.11, 3.12).

Запропонований в [15] алгоритм тасування точок автоматично виключає можливість перенавчання і, незважаючи на відносно погану якість навчання, дає результати при прогнозі на порядок кращі, ніж результати навчання за іншими алгоритмами, що використовують перший підхід.

Застосування спрощеної схеми зміни мутацій у генетичному алгоритмі збільшує швидкість збіжності і не впливає на якість прогнозу.

3.9. Нейронна мережа Хопфілда та її застосування

Відродження інтересу до нейронних мереж пов'язано з роботою Хопфілда (1982 р.). Ця робота пролила світло на ту обставину, що запозичені з природи мережі з нейроноподібних елементів можуть бути використані для обчислювальних цілей. Дослідники з багатьох галузей знань одержали стимул для подальших досліджень цих мереж, переслідуючи при цьому двояку мету: краще розуміння того, як працює мозок, та застосування мозкоподібних властивостей цих мереж для вирішення проблем, що не піддаються вирішенню традиційними методами.

3.9.1. Ідея рекурентності

Нейронна мережа Хопфілда – це приклад мережі, яку можна визначити як динамічну систему з ОС, де вихід однієї цілком прямої операції служить входом наступної операції мережі (рис. 3.13) [22].

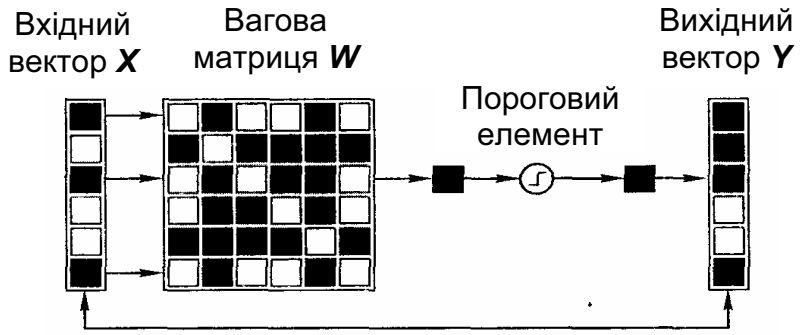


Рис. 3.13. Принципова схема виконання прямої ітерації

Мережі, що працюють як системи зворотного зв'язку, називаються *рекурентними мережами.* Кожна пряма операція мережі називається ітерацією. Рекурентні мережі, подібно до будь-яких інших нелінійних динамічних систем, здатні виявляти цілий спектр різних поведінок. Зокрема, один із можливих зразків поведінки – це те, що система може бути стійкою, тобто вона може сходитися до єдиної фіксованої (нерухомої) точки. Коли нерухома точка є входом у таку динамічну систему, то на виході будемо мати ту ж саму точку. Таким чином, система залишається зафіксованою в тому ж самому стані. Можливі періодичні цикли або хаотична поведінка.

Було показано, що мережі Хопфілда стійкі. У загальному випадку може бути більше однієї фіксованої точки. Те, до якої фіксованої точки буде сходитися мережа, залежить від вихідної точки, обраної для початкової ітерації.

Нерухомі точки називаються *атракторами.* Множина точок (векторів), що притягаються до визначеного атрактору в процесі ітерацій мережі, називається областю притягання цього атрактора. Множина нерухомих точок мережі Хопфілда – це її *пам'ять.* У цьому випадку мережа може діяти як асоціативна пам'ять. Ті вхідні вектори, що попадають у сферу притягання окремого атрактора, є зв'язаними (асоційованими) з ним.

Наприклад, атрактор може бути деяким бажаним образом. Область притягання може складатися із зашумлених або неповних версій цього образу. Є надія, що образи, які неясно нагадують бажаний образ, будуть згадані мережею як асоційовані *з* даним образом.

3.9.2. Бінарні мережі Хопфілда

На рис. 3.13 зображено схему обробки інформації в мережі Хопфілда. Вхідні і вихідні вектори складаються з «–1» і «4–1» (замість «–1» можна використати «0»). Є симетрична вагова матриця, що складається із цілих чисел з нулями по діагоналі $\boldsymbol{W} = \|\boldsymbol{w_{ij}}\|$. Вхідний вектор $\boldsymbol{X}$ помножується на вагову матрицю, використовуючи звичайне матрично-векторне множення. Однак тільки одна компонента вихідного вектора $\boldsymbol{Y} = \|\boldsymbol{y_{ij}}\|$ використовується на кожній ітерації. Ця компонента, що може бути обрана випадково, подається на пороговий елемент, чий вихід або –1, або +1. Відповідна компонента вхідного вектора заміняється на це значення і таким чином утворюється вхідний вектор для наступної ітера-

ції. Ця процедура відома як «асинхронна корекція» [22]. Процес продовжується доти, доки вхідні і вихідні вектори не стануть однаковими (тобто поки не буде досягнута нерухома точка). Цей алгоритм описано нижче.

3.9.3. Опис алгоритму

1. Обчислити компоненти вихідного вектора $y_j, j = \overline{1,n}$, за формулою

$$y_j = T\left(\sum_{i=1}^{n} w_{ij} x_i\right), \tag{3.53}$$

де $T(x) = \begin{cases} -1, \text{ якщо } x < 0; \\ 1, \text{ якщо } x > 0. \end{cases}$

2. Виконати асинхронну корекцію:

а) почати з вхідного вектора $(x_1, x_2, \ldots, x_n)$;

б) знайти y_j відповідно до формули (3.53);

в) замінити $(x_1, x_2, \ldots, x_n)$ на $(y_1, x_2, x_3, \ldots, x_n) = Y$ і подати на вхід;

г) повторити процес, щоб знайти y_2, y_3 *і* т.д. (y_j можуть вибиратися випадково).

3. Повторювати кроки б–в доти, доки вектор $Y = (y_1, y_2, \ldots, y_n)$ не перестане змінюватися. Кожен крок зменшує величину енергії зв'язків

$$E = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} x_i x_j \tag{3.54}$$

так, що забезпечує збіжність до нерухомої точки (атрактору).

Асинхронна корекція й нулі на діагоналі матриці W гарантують, що енергетична функція (3.54) буде зменшуватися з кожною ітерацією. Асинхронна корекція особливо істотна для забезпечення збіжності з нерухомою точкою. Якщо ми допустимо, щоб весь вектор коригувався на кожній ітерації, то можна одержати мережу з періодичними циклами як термінальними станами, а не нерухомими точками.

3.9.4. Зразки поведінки

Саме вагова матриця відрізняє поведінку однієї мережі Хопфілда від іншої, так що виникає питання: «Як визначити цю вагову матрицю?». Відповідь: треба задати визначені вагові вектори, що називають *екземплярами. Є* надія, що ці екземпляри будуть фіксованими точками результуючої мережі Хопфілда. Хоча це не завжди так. Для того щоб екземпляри були атракторами, вагову матрицю $W = \|w_{ij}\|$ треба задати так [22, 32]:

$$w_{ij} = \begin{cases} \sum_{k=1}^{N} (x_{k_i} - 1)(x_{k_j} - 1) \text{ для } i \neq j; \\ 0 \text{ для } i = j, \end{cases} \tag{3.55}$$

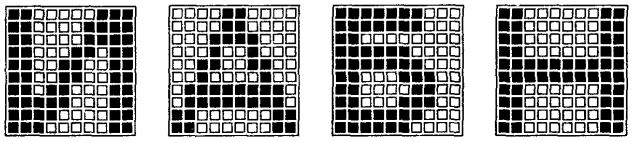
де N –- кількість заданих екземплярів; $\boldsymbol{X}_k = \{\boldsymbol{x}_{k_1}, \boldsymbol{x}_{k_2}, \ldots, \boldsymbol{x}_{k_n}\}$ – k-й екземпляр.

Якщо екземпляри векторів утворять множину ортогональних векторів, то можна гарантувати, що коли вагова матриця вибирається як показано вище, то кожен екземпляр вектора буде нерухомою точкою. Однак у загальному випадку, для того щоб екземпляри приводили до нерухомих точок, ортогональність не обов'язкова.

3.9.5. Застосування мережі Хопфілда

Мережа Хопфілда може використовуватися, зокрема, для розпізнавання образів. Але кількість розпізнаваних образів не занадто велика внаслідок обмеженості пам'яті в мережах Хопфілда. Далі наведено результати дослідження її роботи при навчанні на чотирьох літерах російського алфавіту.

Вихідні образи:



Дослідження проводилося таким чином: послідовно збільшуючи зашумленість кожного з чотирьох образів, вони подавалися на вхід мережі Хопфілда. Результати роботи мережі ретельно розглянуто у [34].

За результатами експериментів можна зробити висновок, що НМ Хопфілда чудово справляється із задачею розпізнавання образів для експериментів з перекручуванням на 0...40 %. У цьому діапазоні всі еталони розпізнаються без помилок (іноді виникають незначні перешкоди для 40 % зашумлення).

При 45...60 % зашумлення образи розпізнаються нестабільно, часто виникає «переплутування» і на виході НМ з'являється зовсім інший еталон або його негатив.

Починаючи з 60 % зашумлення на виході системи буде з'являтися негатив образа, що тестується, іноді частково перекручений (при 60...70 %).

3.9.6. Ефект «перехресних асоціацій»

Ускладнимо задачу і навчимо нашу НМ Хопфілда ще одному зразку, а саме літері «П».

Літера «П» дуже схожа на вже існуючі в пам'яті НМ літери «И» та «Н» (рис. 3.14, а). Тепер НМ Хопфілда не може розпізнати ні одну з цих літер навіть у неспотвореному стані. Замість правильно розпізнаної літери вона видає зображення, показане на рис. 3.14, б (при зашумленні образа від 0 до 50 %). Воно схоже потроху на кожну з літер «И», «Н», «П» і не є правильною інтерпретацією жодної з них.

При зашумленні від 50 до 60 % на виході НМ спочатку з'являється наведене зображення у злегка перекрученому вигляді.

Починаючи з 65 % зашумленості, на виході НМ стабільно з'являється негатив зображення (рис. 3.14, в).

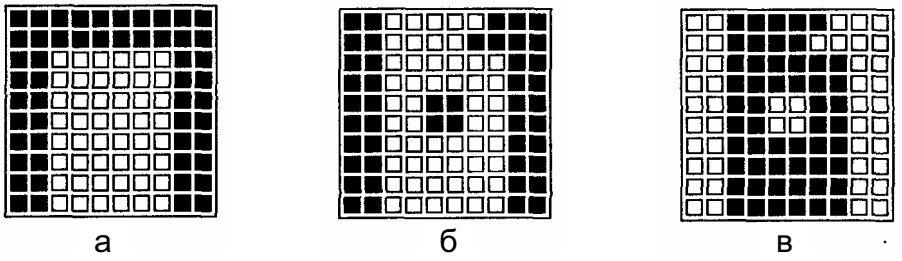


Рис. 3.14. Ілюстрація ефекту перехресних асоціацій

Описане функціонування нейронної мережі відоме як ефект «перехресних асоціацій». При цьому символи «А» і «Б» розпізнаються безпомилково при перекручуванні до 40 %. При 45...65 % на виході НМ з'являються трохи зашумлені інтерпретації зображення, схожі на негатив літери «Б» (але дуже перекручений), або ж негатив образу, що тестується. При перекручуванні 70 % і більше НМ стабільно розпізнає в образі, що тестується, його негатив.

3.10. Нейронна мережа Хеммінга

Коли немає потреби, щоб мережа в явному вигляді давала зразок, тобто достатньо, скажімо, отримати номер зразка, асоціативну пам'ять успішно реалізує мережа Хеммінга [22]. Дана мережа характеризується, порівняно з мережею Хопфілда, меншими витратами на пам'ять та об'ємом обчислень, що стає очевидним з її структури (рис. 3.15).

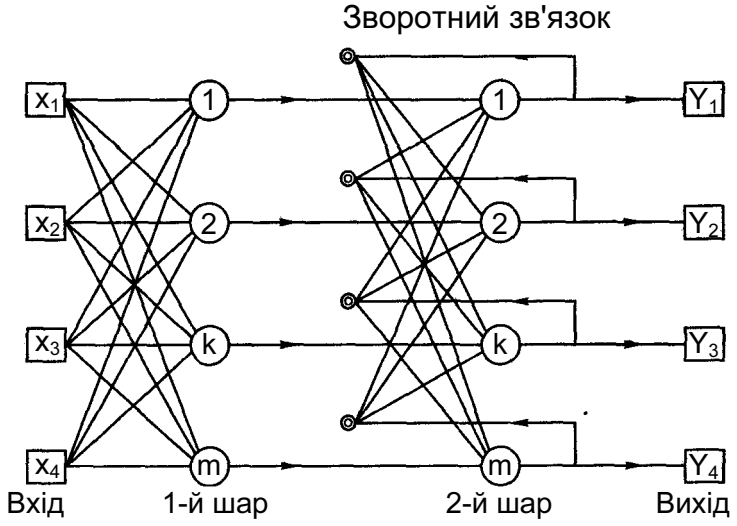


Рис. 3.15. Структурна схема мережі Хеммінга

Мережа складається з двох шарів. Перший і другий шари мають по ***m*** нейронів, де ***m*** – кількість зразків. Нейрони першого шару мають по ***n*** синапсів, з'єднаних входами мережі (що утворюють фіктивний нульовий шар). Між нейронам другого шару існують інгібіторні (негативно зворотні) синаптичні зв'язки. Єдиний синапс з позитивним зворотним зв'язком для кожного нейрона з'єднаний з його ж аксоном.

Ідея роботи мережі полягає в пошуку відстані Хеммінга від зразка, що тестується, до всіх інших зразків.

Відстанню Хеммінга називається кількість відмінних бітів у двох бінарних векторах. Мережа повинна вибрати зразок з мінімальною відстанню Хеммінга до невідомого вхідного сигналу, в результаті чого буде активізовано тільки один вихід мережі, що відповідає цьому зразку.

На стадії ініціалізації вагових коефіцієнтів першого шару та порогу активаційної функції їм будуть присвоєні такі значення:

$$w_{ij} = \frac{x_i^k}{2}, i = 0 \ldots n-1, k = 0 \ldots m-1;$$

$$T_k = \frac{n}{2}, k = 0 \ldots m-1,$$

де x_i^k – i-й елемент k-го зразка.

Вагові коефіцієнти гальмуючих синапсів у другому шарі беруть такими, що дорівнюють величині $0 < h < \frac{1}{m}$. Синапс нейрона, зв'язаний з його ж аксоном, має вагу +1.

3.10.1. Алгоритм функціонування мережі Хеммінга

Алгоритм функціонування мережі Хеммінга має такі кроки:

1. На входи мережі подається невідомий вектор, виходячи з чого розраховуються стани нейронів першого шару (верхній індекс шару в дужках показує номер шару):

$$y_j^{(1)} = f\left(s_j^{(1)}\right) = f\left(\sum_{i=0}^{n-1} w_{ij} x_i + T_j\right), j = 0 \ldots m-1.$$

Після цього отриманими значеннями ініціалізуються значення аксонів другого шару:

$$y_j^{(2)} = y_j^{(1)}, j = 0 \ldots m-1.$$

2. Обчислюються нові значення входів нейронів другого шару:

$$s_j^{(2)}(p+1) = y_j(p) - \varepsilon \sum_{k=0}^{m-1} y_k^{(2)}(p),$$

$$k \neq j, j = 0 \ldots m-1$$

та значення їх аксонів

$$y_j^{(2)}(p+1) = f\left| s_j^{(2)}(p+1) \right|, j = 0 \ldots m-1.$$

Активаційна функція f має вигляд порога, при цьому величина порога має бути достатньо великою, щоб будь-які можливі значення аргументу не приводили до насичення.

3. Перевіряється, чи змінились виходи нейронів другого шару за останню ітерацію. Якщо так – перейти до кроку 2. Інакше – закінчення.

З опису алгоритму видно, що роль першого шару досить умовна: використавши один раз на кроці 1 значення його вагових коефіцієнтів, мережа більше не звертається до нього, тому перший шар може бути взагалі виключений з мережі (замінений на матрицю вагових коефіцієнтів).

Переваги цього алгоритму:

– невеликі витрати на пам'ять;
– мережа працює гранично швидко;
– надзвичайно простий алгоритм роботи;
– ємність мережі не залежить від розмірності вхідного сигналу (як в НМ Хопфілда) і точно дорівнює кількості нейронів.

3.10.2. Аналіз результатів експериментів

Наведемо результати експериментів з НМ Хеммінга в задачі розпізнавання літер російського алфавіту.

НМ Хеммінга чудово справляється з задачею розпізнавання образів для експериментів з рівнем шумів від 0 до 45 %. У цьому діапазоні всі еталони розпізнаються без помилок.

При 50 %-му зашумленні образи розпізнаються нестабільно, часто виникає «переплутування» і на виході НМ з'являється зовсім другий еталон.

При 80–90 %-му зашумленні образ починає інвертуватися і при зашумленні 100 % бачимо повністю інвертоване початкове зображення. Хоча людина може розпізнати з легкістю інвертований символ, програма не вміє цього робити, бо вважає, що сам символ зображується тільки чорними точками. З цікавості можна було б навчити мережу розпізнавати інвертовані зображення символів одночасно з неінвертованими. Для цього необхідно процес розпізнавання розбити на два етапи: спочатку НМ працює в нормальному порядку і обчислює виходи другого шару для оригінального (можливо, зашумленого) зображення. Потім оригінальне зображення інвертується і процес розпізнавання ініціюється знову. Нарешті максимальні значення виходів, які вибрані з двох випробувань, порівнюються та вибирається найбільше.

При зашумленні 80...100 % НМ Хеммінга переплутує символи, що подаються, з найбільш схожими еталонними. Наприклад, при 100-відсотковому зашумленні літера «П» схожа на товсту смугу і програма розпізнає символ як «1».

Теоретично, другий шар повинен працювати, поки його виходи не стабілізуються, але на практиці кількість ітерацій штучно обмежують, так було зроблено і в наведених експериментах.

Мережі зі зворотними зв'язками є перспективним об'єктом подальших досліджень, їх динамічна поведінка відкриває нові цікаві можливості.

Зрештою, можна зробити таке узагальнення. Мережі Хопфілда та Хеммінга дозволяють просто та ефективно вирішувати задачу відтворення образів з неповної та перекрученої інформації. Невелика ємність мереж (кількість образів, що запам'ятовуються) пояснюється тим, що мережі не просто запам'ятовують образи, а й дозволяють проводити їх узагальнення, наприклад, за допомогою мережі Хеммінга можлива класифікація за критерієм максимальної правдоподібності. Разом з тим, легкість побудови програмних та апаратних моделей роблять ці мережі привабливими для багатьох застосувань [22, 32].

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Кокорева Л.В., Перевощикова О.Л., Ющенко К.Л. Діалогові системи та представлення знань – К.: Наук. думка, 1992. – 444 с.
2. Шостак И.В. Управление сложными объектами в реальном времени на основе динамических экспертных систем // Авиационно-космическая техника и технология. – Х.: ГАКУ „ХАИ”. – 1999. – Вып. 10. – С. 204–210.
3. Шостак И.В. Методы ускорения вывода на знаниях в динамических экспертных системах реального времени // Радиоэлектроника и информатика. – 1999. – № 3. – С. 23–26.
4. Шостак В.Ф., Шостак И.В. Динамические экспертные системы управления объектами с неполной информацией // Теория и техника передачи, приема и обработки информации (Новые информационные технологии). – Х.: ХТУРЭ. – 2000. – С. 211–213.
5. Шостак И.В. Текущее состояние базы знаний в динамических экспертных системах управления сложными объектами // Радиоэлектроника и информатика. – 2000. – № 3. – С. 68–71.
6. Шостак В.Ф. Модели и методы управления сложными технологическими комплексами в нештатных (экстремальных) режимах работы в АСУ ТП // Автоматика и телемеханика. – М.: РАН. – 1994. – № 10. – С. 158–164.
7. Букатова И.А., Михасев Ю.И., Шаров А.М. Эвоинформатика. Теория и практика эволюцинного моделирования. – М.: Наука, 1991. – 206 с.
8. Васильев В.И. Распознающие системы: Справочник. – К.: Наук. думка, 1983. – 298 с.
9. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности / Г.К. Вороновский, К.В. Махотило, С.Н. Петрашев, С.А.Сергеев – Х.: Основа, 1997. – 112 с.
10. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. – СПб.: Питер, 2000. – 384 с.
11. Журавлев Ю.И., Гуревич И.Б. Распознавание образов и распознавание изображений // Распознавание, классификация, прогноз. Математические методы и их применение. – М.: Наука. – 1989. – Вып. 2. – С. 5 – 72.
12. Заенцев И.В. Нейронные сети: основные модели. – Воронеж: ВГУ, 1999. – 157 с.
13. Зайченко Ю.П., Заєць І.О. Синтез і адаптація нечітких прогнозуючих моделей на основі методу самоорганізації // Наукові вісті НТУУ „КПІ”. – 2001. – № 3. – С. 34–41.
14. Зайченко Ю.П., Заєць І.О. Застосування рекурсивних методів ідентифікації в задачах синтезу нечітких прогнозуючих моделей // Праці Міжнар. конф. з індуктивного моделювання. Львів, травень 2002. – С. 59–64.
15. Зайченко Ю.П., Кебкал О.Г., Крачковський В.Ф. Нечіткий метод

групового урахування аргументів та його застосування в задачах прогнозування макроекономічних показників // Наукові вісті НТУУ „КПІ”. – 2000. – № 2. – С. 18–26.

16. Зайченко Ю.П. Нечіткий метод індуктивного моделювання в задачах прогнозування макроекономічних показників // Системні дослідження й інформаційні технології. – 2003. – № 3. – С. 25–45.

17. Ивахненко А.Г. Непрерывность и дискретность. Переборные методы моделирования и кластеризации. – К.: Наук. думка, 1990. – 224 с.

18. Ивахненко А.Г., Юрачковский Ю.П. Моделирование сложных систем по экспериментальным данным. – М.: Радио и связь, 1987. – 118 с.

19. Ивахненко А.Г., Мюллер Й.А. Самоорганизация прогнозирующих моделей. – К.: Техніка, 1985. – 223 с.

20. Ивахненко А.Г., Зайченко Ю.П., Димитров В.Д. Принятие решений на основе самоорганизации. – М.: Сов. радио, 1976. – 280 с.

21. Искусственный интеллект: Справочник – В 3 кн. / Под ред. Д. Поспелова. – М.: Радио и связь, 1990. – Кн. 2. Модели и методы. – 304 с.

22. Короткий С. Нейронные сети Хопфилда и Хэмминга. Опубликовано в Internet. http://www.acdwp.narod.ru/ot3.htm

23. Короткий С. Нейронные сети: алгоритм обратного распространения. Опубликовано в Internet. http://www.acdwp.narod.ru/ot3.htm

24. Короткий С. Нейронные сети: обучение без учителя. Опубликовано в Internet. http://www.acdwp.narod.ru/ot3.htm

25. Кохонен Т. Ассоциативная память. — М.: Мир, 1980. – 235 с.

26. Куссуль Э. М. Ассоциативные нейроподобные структуры. – К.: Техніка. – 1992. – 210 с.

27. Минский М., Пайперт С. Перцептроны: Пер.с англ. – М.: Мир, 1971. – 261 с.

28. Нильсен И. Принципы искусственного интеллекта: Пер.с англ. – М.: Наука, 1985. – 373 с.

29. Осовский С. Нейронные сети для обработки информации / Пер. с польск. И.Д. Рудинского. – М.: Финансы и статистика, 2002. – 344 с.

30. Терано Т., Асаи К., Сугэно М. Прикладные нечеткие системы. – М.: Мир, 1993. – 230 с.

31. Сотник С.Л. Основы проектирования систем искусственного интеллекта: Конспект лекций. 1998. Опубликовано в Internet. http://stratum11.pstu.ac.ru/~leonid/base/intelect.zip.

32. Уосермен Ф. Нейрокомпьютерная техника. Теория и практика: Пер. с англ. – М.: Мир, 1992. – 240 с.

33. Шлезингер М.И. Математические средства обработки изображений. – К.: Наук. думка, 1989. – 200 с.

34. Зайченко Ю.П. Основи проектування інтелектуальних систем: Навч. посібник. – К.: Вид. дім «Слово», 2004. – 352 с.

Шостак Ігор Володимирович
Топал Олексій Сергійович
Устінова Олександра Миколаївна

ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ КЕРУВАННЯ СКЛАДНИМИ ОБ'ЄКТАМИ

Редактор О.Ф. Серьожкіна

Зв. план, 2005
Підписано до друку 04.11.2005
Формат 60x84 1/16. Папір офс. № 2. Офс. друк
Ум. друк. арк. 4. Обл.-вид. арк. 4,5. Наклад 100 прим. Замовлення 512.
Ціна вільна

Національний аерокосмічний університет ім. М.Є. Жуковського
«Харківський авіаційний інститут»
61070, Харків-70, вул. Чкалова, 17
http://www.khai.edu
Видавничий центр «ХАІ»
61070, Харків-70, вул. Чкалова, 17
izdat@khai.edu