ДЕРЖАВНА ФІСКАЛЬНА СЛУЖБА УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ДЕРЖАВНОЇ ПОДАТКОВОЇ СЛУЖБИ УКРАЇНИ

СЕРІЯ «ПОДАТКОВА ТА МИТНА СПРАВА В УКРАЇНІ»

Серію «Митна справа в Україні» засновано в 2001 р. Її продовженням є серія «Податкова та митна справа в Україні», яку започатковано в 2015 р.

В. М. Антоненко,
С. Д. Мамченко,
Ю. В. Рогушина

СУЧАСНІ ІНФОРМАЦІЙНІ СИСТЕМИ І ТЕХНОЛОГІЇ: УПРАВЛІННЯ ЗНАННЯМИ

Навчальний посібник

Ірпінь
2016

УДК 004.4(075.8)
ББК 32.97я73
А72

Рекомендовано до друку Вченою радою
Національного університету ДПС України
(протокол №11 від 23 квітня 2015 р.)

Рецензенти:
Устенко Станіслав Веніамінович, д.е.н., професор, завідувач кафедри інформаційних систем в економіці КНЕУ ім. Вадима Гетьмана;
Ріппа Сергій Петрович, д.е.н., професор, декан факультету інформаційних технологій та менеджменту НУДПСУ.

Антоненко В. М.

А72 Сучасні інформаційні системи і технології: управління знаннями : навчальний посібник / В. М. Антоненко, С. Д. Мамченко, Ю. В. Рогушина. – Ірпінь : Національний університет ДПС України, 2016. – 212 с. – (Серія «Податкова та митна справа в Україні», т. 52).

ISBN 978-966-337-418-5

Навчальний посібник присвячений сучасним інформаційним системам і технологіям управління знаннями. Він містить теоретичні відомості, які можна використати у лекційних заняттях, та лабораторні роботи для набуття практичних навичок і закріплення теоретичних знань. У ньому сформуються базові поняття щодо категорії «знання», дається класифікація знань та моделі подання знань в інтелектуальних системах, визначається сутність системи управління знаннями та основних інформаційних технологій управління ними, розглядається формалізація знань як нечітка система. Для набуття практичних навичок пропонується застосування технологій системи MatLab.

Призначений для магістрів, аспірантів, викладачів, які досліджують технології управління знаннями.

УДК 004.4(075.8)
ББК 32.97я73

ISBN 978-966-337-418-5

ЗМІСТ

ВСТУП

Становлення і розвиток інформаційного суспільства є характерною рисою XXI століття. Саме в інформаційному суспільстві активно розвиваються інформаційні і комунікаційні технології, створюються умови для ефективного використання знань у вирішенні найважливіших завдань управління суспільством і демократизації суспільного життя. Світова спільнота, ставши на шлях постіндустріальної цивілізації становлення і розвитку інформаційного суспільства, формує різні шляхи його побудови.

Характерними рисами й ознаками інформаційного суспільства є:

– формування єдиного інформаційно-комунікаційного простору країни як частини світового інформаційного простору, повноправна участь України в процесах інформаційної і економічної інтеграції регіонів, країн і народів;

– становлення і в подальшому домінування у різних сферах перспективних інформаційних технологій, засобів обчислювальної техніки і телекомунікацій;

– створення і розвиток ринку інформації і знань як факторів виробництва на додаток до ринків природних ресурсів, праці й капіталу, перехід інформаційних ресурсів суспільства в реальні ресурси соціально-економічного розвитку, фактичне задоволення потреб суспільства в інформаційних продуктах і послугах;

– зростання ролі інформаційно-комунікаційної інфраструктури в системі суспільного виробництва;

– підвищення рівня освіти, науково-технічного і культурного розвитку за рахунок розширення можливостей систем інформаційного обміну на міжнародному, національному і регіональному рівнях і, відповідно, підвищення ролі кваліфікації, професіоналізму і здібностей до творчості як найважливіших характеристик послуг праці;

– створення ефективної системи забезпечення прав громадян і соціальних інститутів на вільне одержання, поширення і використання інформації як найважливішої умови демократичного розвитку.

Інформаційне суспільство вносить істотні якісні зміни і в методологію сучасної освіти: знання формують основу для реалізації потреб освіти і складають один з найважливіших виробничих чинників. В інформаційному суспільстві економіка країни зазнає зміни і трансформується в економіку, засновану на інформації і знаннях, інтелектуальних інформаційних технологіях, і підтримується всесторонньо розвиненою людиною. Ці тенденції в економіці вимагають нових видів освіти, етичної компетентності й інфраструктури, що підтримує процес.

Інтелектуальні інформаційні системи повинні мати механізми адаптації, які б дозволяли їм вирішувати поставлені перед ними задачі на основі аналізу поточної ситуації. Людина не отримує всі свої знання і уміння зразу, тим більше неможливо передати всі необхідні знання і уміння інтелектуальній системі, призначеній для вирішення досить складних задач. Тому винятково важливе значення має здатність інтелектуальної системи до *навчання,* тобто до поповнення своїх знань і до набуття нових умінь. Таке навчання може відбуватися шляхом спілкування інтелектуальної системи з людиною або з іншою інтелектуальною системою. Якщо ж навчання відбувається самостійно, йдеться про *самонавчання*.

Чим більш інтелектуальною є система, тим більш ефективно вона може досягати поставленої перед нею мети.

Отже, перелік інтелектуальних задач повинен випливати із загальної концепції розуму, який повинен орієнтуватися в навколишній ситуації, ініціювати якісь дії з метою досягнення своєї мети та накопичувати знання про світ. До цього переліку можна віднести такі задачі:

- аналіз ситуацій;
- розпізнавання образів;
- логічне мислення;
- розуміння нової інформації;
- навчання і самонавчання;
- планування цілеспрямованих дій.

На основі вищесказаного можна сформулювати таке визначення інтелектуальної системи.

Інтелектуальною системою називається самокерована кібернетична система, яка має певну суму знань про світ і здатна на основі безпосереднього сприйняття і подальшого аналізу поточної ситуації планувати дії, спрямовані на досягнення певної мети, а також поповнювати свої знання.

Цей навчальний посібник присвячений сучасним інформаційним системам і технологіям, що базуються на знаннях.

ТЕМА 1
БАЗОВІ ПОНЯТТЯ УПРАВЛІННЯ ЗНАННЯМИ

1. Інформація та знання в сучасних наукових дослідженнях.
2. Інформація, дані та знання в інтелектуальних системах.
3. Особливості знань, що відрізняють їх від даних.
4. Типи знань в інтелектуальних системах.
5. Важливість управління знаннями.
6. Підходи до управління знаннями.
7. Поняття про моделі представлення знань в інтелектуальних системах.

1. Інформації та знання у сучасних наукових дослідженнях

Як багатоаспектна загальнонаукова категорія інформація займає вагоме місце у наукових дослідженнях представників як природничих, так і гуманітарних дисциплін. При цьому звертається увага на єдність двох відносно самостійних аспектів, що становлять зміст феномену інформації:

– комунікаційного, пов'язаного з процедурою передачі повідомлення, трансляції відомостей (дослідження комунікацій, комунікаційних технологій, інформаційних мереж тощо);

– змістового, пов'язаного з передачею продуктів інтелектуальної діяльності та їх засвоєнням, спрямованим на зростання обізнаності або зниження невизначеності отримувачів.

У сучасній літературі сформувалось два підходи до визначення сутності інформації: *атрибутивний*, за якого інформація кваліфікується як невід'ємна ознака будь-якого матеріального об'єкта та оцінюється як міра впорядкованості структур та взаємодій; *функціональний*, прихильники якого пов'язують інформацію з функціонуванням та розвитком систем, що самоорганізуються.

Таким чином, сучасні технології збирання, накопичення, зберігання та передачі інформації є лише моментом загального інформаційного процесу, який залишиться незавершеним, якщо отриманий інтелектуальний продукт не буде засвоєний споживачем.

В умовах інтелектуальної економіки (економіки знань) інформація виступає як особливий об’єкт договірних відносин, пов’язаних з її пошуком, добором, збереженням, переробкою, поширенням та використанням у різних сферах людської діяльності. Як специфічне економічне благо, вона є важливим засобом отримання додаткових вигод за рахунок зниження невизначеності та ризику суб’єктів господарювання. Обертаючись у ринковій економіці як товар (інформаційний продукт, послуга) або ресурс, який використовується у процесі господарської діяльності, інформація має певні особливості, а саме: є нематеріальним благом (ідеальним компонентом буття) і не зводиться до фізичних об’єктів, які є її носіями; характеризується невичерпністю, не зменшується у процесі використання та не споживається у традиційному розумінні цього терміна; не локалізована в просторі, легко поширюється, тиражується (за сучасного рівня технологій) і змінює форми фіксації; не зникає у процесі споживання, але для отримання корисного ефекту від останнього потребує певних інтелектуальних навичок; практично не підлягає фізичному зносу, але може морально застарівати; не існує монополії на володіння та використання інформації за винятком тієї її частини, яка є об’єктом інтелектуальної власності.

Необхідно зазначити, що унікальність інформації зумовлена закладеною в ній дихотомією поширеності та рідкісності, невичерпності та скінченності. Незважаючи на те, що в умовах ринкової економіки інформація може виступати об’єктом власності та обміну, саме право власності на інформацію не лише не суперечить можливості її максимального поширення, а й передбачає останню як джерело зростання доходу власника.

Сучасний соціально-економічний розвиток характеризується зростанням ролі та значення факторів, пов’язаних з інтелектуалізацією та інформатизацією господарського життя суспільства.

Знання – результат пізнавальної діяльності людини, об’єктивізований знаковими засобами мови. Знання мають інформаційну природу та характеризуються ознаками продуктивності, структурованості та латентності. Перетворення інформації на знання є результатом інтелектуальної діяльності людини. Відтак інтелект

у широкому розумінні – здатність генерувати та накопичувати знання.

Знання – найзагальніше вираження для позначення теоретичної діяльності розуму, що має претензію на об'єктивну істину (на відміну від мислення чи думки, які можуть бути явно фантастичними). Терміни «знання» і «пізнання» належать, по суті, до одного і того ж предмета, відрізняються певним відтінком: перший стосується більше об'єктивного аспекту і результатів розумового процесу, другий – більше його суб'єктивних умов.

У сучасній науковій літературі розрізняють також:

– *декларативні знання*, які пов'язані з описом та інформацією, що уможливлюють відповідь на питання «що робити?»;

– *процедурні знання*, які застосовуються для здійснення нових дій і зумовлюють відповідь на питання «як діяти?»;

– *епізодичні знання*, які використовуються на основі подібності ситуацій, аналогічності проблем, подій, передумов;

– *евристичні знання*, які належать до практичних правил, виведених на основі власного досвіду;

– *метазнання*, пов'язані з розумінням способів та методів пошуку й переробки інформації та прийняття рішень.

Виокремлюють також *організаційні знання*, які включають принципи, факти, навички, правила та методи, що забезпечують кадровий потенціал та ділову активність певної організації. У структурі організаційних знань виокремлюють *теоретичні, практичні, стратегічні, комерційні, виробничі* та інші знання.

Знання також поділяють на *інтенсіональні* та *екстенсіональні*. *Інтенсіональні* знання являють собою понятійні, концептуальні знання щодо класів об'єктів предметної області та їх відношень; *екстенсіональні* – це кількісні характеристики інтенсіональної частини, тобто це база даних автоматизованої експертної системи.

Сучасні знання ставлять вимогу розрізняти: *кодифіковані знання*, які можна виокремити, систематизувати, зафіксувати, скопіювати, передати, забезпечити їм правову охорону; *некодифіковані знання*, які не можна повністю зафіксувати, важко виокремити, захистити і проконтролювати їх використання.

Кодифіковане знання може бути представлене у формі усних або письмових текстів, баз даних, картотек, символьної або графічної інформації тощо. Для збереження та передачі такого знання використовують різноманітні матеріальні носії (паперові, аудіо-, відео-, електронні). Важливо зазначити, що кодифікація знань виконує такі функції: *креативну* (творчу), спрямовану на виявлення теоретичних та методологічних недоліків, слабких місць у логіці аргументації тощо; *комунікативну*, спрямовану на поширення знань, полегшення доступу до них з боку суспільства; *культурну*, спрямовану на накопичення та збереження інтелектуальних скарбів людства; *економічну*, спрямовану на надання знанням форми, в якій останні можуть стати об'єктом інтелектуальної власності, товаром, капіталом.

Аналіз показав, що в сучасній економічній літературі є різні підходи до класифікації знань. Так, деякі дослідники розрізняють *формалізовані* (записані на певному носієві: папері, відео-, аудіо-, електронному) та *неформалізовані* (ті, що зберігаються у людській пам'яті) знання. Відповідно до концепції М. Поланьї існують *явні* (точно визначені, збережені, експліковані логіко-вербальними формами) та *неявні* (такі, що повністю не вербалізуються, не вилучаються та чітко не ідентифікуються) знання. Відомий західний дослідник К. Вінг структурує знання відповідно до чотирьох концептуальних рівнів, виокремлюючи: *знання щодо мети* (світогляд, цілі, система понять); *систематичні знання* (знання систем, схем, методів); практичні знання (пов'язані з умінням приймати рішення); *автоматичні знання*, які застосовуються без залучення логічних міркувань.

За рівнем системності розрізняють: *інтуїтивні знання* (асоціації, судження, гіпотези); *понятійні знання* (тезауруси, фрейми, семантичні мережі); систематизовані знання (правила, теореми, закони).

Досліджуючи специфіку інтелектуальної економіки, важливо розрізняти *процеси генерації* та *об'єктивізації знань*.

Генерація знань – процес «еманації», творення нового знання шляхом переробки інформації. Джерелом генерації нового знання є інтелектуальна активність: заснована на знаннях свідома

діяльність людини, здатної накопичувати, засвоювати та творчо опрацьовувати інформацію.

Об'єктивізація знань – процес перетворення знання із суб'єктивної в об'єктивну форму, пов'язану з різними матеріальними носіями (технічне обладнання, документація, патенти, бази даних, програми, книги тощо). Об'єктивізація знань не означає їх уречевлення, оскільки, втілившись у річ, матерію, знання, залишаються за своєю природою ідеальним утворенням.

Знання вплетені в соціальну структуру суспільства, вони поєднують інформацію та ціннісні установки, здатність людей засвоювати смислові моделі. Як феномен культури, знання характеризується певними закономірностями розвитку, пов'язаними як із загальним процесом пізнання, так і з виробленими культурою формами організації та осмислення дійсності.

Загальновизнано, що у постіндустріальному суспільстві знання, інформація виступають не просто втіленою у засоби виробництва субстанцією, а безпосередньою продуктивною силою, стратегічним ресурсом, імпульсом приведення в дію надзвичайно потужних систем, тим ферментом, який, вивільняючи додаткову енергію, знижує ентропію мікро- та макросистем. Відтак, аналізуючи інформацію та знання як нову продуктивну силу, слід зазначити не просто про сукупність відомостей, даних, а про нові можливості, що втілені у людині, яка володіє інформацією, засвоїла її, примноживши таким чином свої знання.

2. Інформація, дані та знання в інтелектуальних системах

Для розробки систем штучного інтелекту – інтелектуальних систем (експертних систем, систем підтримки прийняття рішень та інших) використовуються дані та знання предметної області для розв'язання прикладних задач.

Дані – це окремі факти, що характеризують об'єкт, процеси та явища предметної галузі, а також їх властивості. При розробці, проектуванні інтелектуальних систем дані проходять етапи трансформації від більш узагальнених множин до більш вузьких, конкретизованих множин, необхідних для розв'язку прикладних задач:

1) D_1 – дані як результат вимірювань та спостережень;

2) D_2 – дані на матеріальних носіях інформації (таблиці, протоколи, довідники);

3) D_3 – моделі (структури) даних у вигляді діаграм, графіків, функцій.

4) D_4 – дані в комп'ютері на мові опису даних;

5) D_5 – бази даних на комп'ютеризованих (електронних) носіях інформації.

Знання засновані на даних, що отримані емпіричним шляхом. Вони є результатом інтелектуальної діяльності людини.

Знання – це закономірності предметної галузі (принципи, зв'язки, закони), отримані під час практичної діяльності та професійного досвіду людини.

Нині у штучному інтелекті не існує строго формалізованого визначення поняття «знання». Більшість фахівців, розробників інтелектуальних систем використовують таке визначення: *знання* – це добре структурована інформація, що зберігається в системі і містить усі відомості про предметну область та правила виводу, що необхідні для розв'язку безлічі завдань інтелектуальної системи. Під час проектування та розробки інтелектуальної системи знання проходять аналогічну даним трансформацію – від більш узагальнених множин до більш вузьких, конкретизованих для даної предметної області:

1) R_1 – знання в пам'яті людини як результат мислення;

2) R_2 – знання на матеріальних носіях інформації (довідники, підручники та ін.);

3) R_3 – поле знань – умовний опис основних об'єктів предметної області, їх атрибутів та закономірностей, що їх пов'язують;

4) R_4 – формалізовані знання, що описані на специфічній мові представлення знань;

5) R_5 – бази знань, як елемент прикладної інтелектуальної системи на комп'ютеризованих (електронних) носіях інформації.

Для зберігання даних використовуються бази даних, для зберігання знань – відповідно бази знань. *База знань* – сукупність знань, представлених за допомогою стандартизованих моделей на визначених мовах програмування, які складають основу будь-якої інтелектуальної системи.

З точки зору використання знань для розв'язку прикладних задач, то їх можна розділити на дві категорії: *точні* знання та *евристики.* Перша категорія – це знання, що отримані зі спеціальної літератури, підручників, довідників та ін. Друга категорія – це досвід фахівців у цій предметній області, що накопичений у результаті багаторічної практики. Друга категорія знань має велике значення для підвищення ефективності системи, оскільки дозволяє використовувати у системах суб'єктивні висновки експертів, їх оцінки різних альтернатив з предметної області та ін.

3. Особливості знань, що відрізняють їх від даних

Розглянемо особливості знань, що відрізняють їх від даних. Ці особливості включають такі властивості.

1) *інтерпретованість.* У традиційних обчислювальних програмах дані, що зберігаються у пам'яті комп'ютера, можуть бути інтерпретовані тільки відповідною програмою (специфіка формату даних). Дані без програми не несуть ніякої інформації. Прагнення до того, щоб комп'ютер був спроможний інтерпретувати вміст своєї пам'яті в поняттях, пов'язаних із семантикою розв'язуваної задачі у системах штучного інтелекту, привело до відокремлення знань від даних. Для того щоб система могла «знати», що являє собою та чи інша інформаційна одиниця, що зберігається в її пам'яті, необхідно забезпечити ці одиниці пояснювальними описами. Ці описи передбачають присвоєння даній одиниці знань системи імен. Система імен включає індивідуальне ім'я даної інформаційної одиниці плюс послідовність імен тих множин чи класів, до яких ця одиниця входить. Ця властивість притаманна реляційним базам даних. Наявність цих надлишкових імен дозволяє системі знати, що зберігається в її БЗ;

2) *рекурсивна структурованість*. Для знань виконується «принцип матрьошки», тобто рекурсивне вкладення одних інформаційних одиниць в інші. Між певними одиницями знань можна встановити такі відносини, як «елемент – множина», «тип – підтип», «ситуація – підситуація», що відображають характер їхнього взаємозв'язку. Це дозволяє ефективно використовувати принцип наслідування, тобто в одному екземплярі зберігати інформацію, однакову для елементів класу. Це дає також можливість робити агрегування і декомпозицію інформаційних одиниць через відносини «елемент – множина», «тип – підтип», «рід – вид» та ін.;

3) *зв'язаність.* У БЗ між інформаційними одиницями встановлюються зв'язки різного типу, що характеризують відносини між об'єктами, явищами та ін. Наприклад, зв'язки типу «причина – наслідок», «бути справедливим для», «сприяти» та ін. Зв'язки дозволяють будувати процедури аналізу знань на суміснність, несуперечність та ін., що важко реалізувати при збереженні традиційних масивів даних;

4) *активність.* Поділ інформаційних одиниць, прийнятий у традиційному програмуванні, на дані і програми (команди) привело до того, що *дані пасивні*, а *команди активні*. У системах, заснованих на знаннях, *знання можуть* ініціювати дії. Поява в інформаційній базі нових фактів, установлення нових зв'язків може стати джерелом активності системи. Знання, що представлені у інтелектуальній системі – це деяка модель, формалізоване представлення предметної області. Водночас дані не виконують задачу змістовного моделювання.

4. Типи знань в інтелектуальних системах

Відокремлюють три типи знань в інтелектуальних системах:

– *декларативні знання* – понятійна чи об'єктна модель предметної області. Одиниця знань – *факт* (без явної вказівки, як і коли цей факт використовується);

– *процедурні знання* – функціональна модель предметної області. Одиниця знань – *правило* (дає спосіб обробки знань);

– *керуючі знання* – стратегія виводу, порядок застосування правил (процедури логічного виводу).

5. Важливість управління знаннями

В умовах сучасної економіки, рушієм якої є інформація, організації вбачають набагато більше цінності у своїх інтелектуальних активах, ніж в активах фізичних. Управління знаннями допомагає підтримувати ті знання, якими необхідно поділитися, якщо вони стануть основою для співпраці. Більше того, управління знаннями допомагає організації:

• ***Стимулювати інновації.*** Управління знаннями забезпечує інфраструктуру для побудови електронних і соціальних мереж з метою розвитку нових продуктів або послуг; сприяє і забезпечує доступ до різноманітних ідей, таким чином даючи іншим людям можливість мати від них користь.

• ***Сприяти розвитку співпраці.*** Управління знаннями збільшує можливості співпраці; збагачує процес обміну експліцитними та непрямими знаннями між людьми; також заохочує вільне пересування ідей.

• ***Заохочувати і використовувати можливості навчання.*** Управління знаннями сприяє і прискорює процес навчання; створює можливості для індивідів і груп людей застосовувати отримані знання на практиці; діє як рушій знань організації і доносить потрібну інформацію потрібним людям у зрозумілому контексті, що націлений на вирішення нових питань; надає індивідуальному навчанню особливої цінності, обдаровуючи його гарними результатами і піднімаючи на загальноорганізаційний рівень.

• ***Збільшити соціальний капітал.*** Управління знаннями збільшує обсяг передачі індивідуальних знань організації; сприяє процесу обміну всередині і поза межами організації, в різний час і в різних місцях; також з'єднує людей, що володіють необхідними прямими і непрямими знаннями, з тими, кому такі знання необхідні для виконання робочих обов'язків.

• ***Залучати і зберігати людський капітал.*** Управління знаннями збільшує рівень збереження кадрів шляхом підкреслення цінності знань працівника і винагородження його за це; фіксує та ефективно використовує те, що люди знають, тобто їхні «ноу-хау» («знаю як») і «ноу-вот» («знаю що»); сприяє кар'єрному зростанню.

• ***Створювати і використовувати структурний капітал.*** Управління знаннями перетворює інтелектуальний капітал у структурний капітал і фокусує стратегічне мислення на використання знань на противагу зосередженню на бюджеті.

• ***Відкривати можливість електронного управління.*** Управління знаннями забезпечує інформаційну базу, що заснована на інформаційних технологіях, для надання громадянам і клієнтам можливості доступу до необхідних їм інформації та послуг; також встановлює мережу інтелектуальних урядових працівників, які можуть підвищити якість вимог громадян і клієнтів.

• ***Збільшити продуктивність.*** Управління знаннями раціоналізує операції та скорочує витрати, ризик, криві ефективності навчання і час початку навчання шляхом усунення непотрібних або надмірних процесів; робить прямий внесок у кінцеві цілі/місії.

• ***Ділитися найкращою практикою і процесами.*** Управління знаннями поширює найкраще у практиці у своїй організації; виносить уроки з невдалих спроб; забезпечує базу для багаторазового користування знаннями і впровадження інновацій; встановлює критерії для роботи індивідів і команд всередині компанії і на зовнішньому рівні; також допомагає новим працівникам швидко адаптуватися до культури організації.

• ***Забезпечити лідерство і прийняття рішень.*** Управління знаннями надає необхідну інформацію в контексті, що допомагає процесу прийняття рішень; генерує нові знання в процеси рішень шляхом спільного використання і співпраці з гравцями та організаторами спільної справи у процесах прийняття рішень; використовує інформацію та знання для приведення дій організації у відповідність з її місією та баченням.

• ***Збільшити рівень задоволення клієнтів.*** Управління знаннями покращує обслуговування клієнтів шляхом раціоналізації їх часу відгуку; фокусується на знаннях потреб клієнта в напрямку зусиль організації; також покращує результат і досвід, отриманий клієнтами при використанні послуг, що надаються.

• ***Створити конкурентоспроможну перевагу/ринкову диференціацію.*** Управління знаннями сприяє у досягненні найвищої

якості знань; як бізнес-пропозиції, продукту чи послуги, переносить цінність пропозиції від надання товарів і послуг до надання знань і досвіду про ці товари і послуги; також фокусує колективний інтелект організації на потребах клієнта при досягненні ділових місій/цілей.

У сучасних умовах у більшості ефективних західних чи східних компаній, які будують стратегії розвитку, працюють менеджери зі знань. Сфера інтелектуального капіталу – один з найновіших напрямів теорії та практики сучасного менеджменту, який динамічно розвивається.

На ринку, який швидко змінюється та розвивається, де поглиблюється конкуренція, швидко застарівають технології та продукти, успішними можуть бути тільки ті організації, які створюють нові знання, поширюють їх на всю організацію та швидко втілюють у нові технології чи продукти.

Ще у середині 80-х років XX ст. в економічній науці виникло розуміння зростання ролі знань у створенні додаткової вартості. Японський вчений Тайні Сакайя у книзі «Вартість, створювана знаннями, або історія майбутнього» твердив, що традиційні чинники виробництва вже не визначають ту цінність, яку споживачі визнають за товаром чи послугою. Тобто створення продукту чи послуги стає інтелектуальномістким, а економіка перетворюється в систему, що функціонує на основі обміну знань та їх взаємної оцінки.

Один з теоретиків теорії інтелектуального капіталу Карл Вінг так визначає поняття знання, як складової «істин та уявлень, точок зору та концепцій, суджень та гіпотез, методологій та «ноу-хау»:

«Знання накопичуються, організуються, інтегруються та зберігаються для їх застосування у конкретних ситуаціях та вирішенні проблем».

У загальному випадку управління знаннями розглядається як процес створення умов для виявлення, збереження та ефективного використання знань та інформації для підвищення ефективності діяльності.

З практичного погляду управління знаннями – це створення таких умов, за яких інтелектуальний капітал організації надає змогу розробляти нові знання для успішного виконання стратегій та забезпечення конкурентних переваг.

6. Підходи до управління знаннями

Залежно від наявних елементів у практиці розглядаються різні підходи до управління знаннями, які відображені в таблиці 1.

Таблиця 1

№ з/п	Автор	Визначення	Елементи
1.	*В. Р. Буковіц, Р. Л. Уільямс*	*Управління знаннями – це процес, за допомогою якого організація накопичує багатство, опираючись на свої інтелектуальні чи базовані на знаннях організаційні активи*	*Організаційні процеси, культура та імідж організації, особисті знання персоналу, інтелектуальна власність, банки даних*
2.	*Д. Ж. Скирме*	*Управління знаннями – це чітко окреслене і систематичне управління важливими для організації знаннями та процесами управління ними, а саме: збирання, організація, дифузія, застосування і експлуатація знань з метою досягнення цілей організації*	*Збирання даних, оцінка знань, розповсюдження знань, цілеспрямована експлуатація знань*
3.	*Р. Руглес*	*Управління знаннями – це створення цінності шляхом активнішої підтримки досвіду, пов'язаного з «ноу-хау» та знаннями всередині організації та поза її межами*	*Інтелектуальні активи, створення нових знань*

Продовження таблиці 1

4.	*Fmerican Prouctivity & Quality Center*	*Управління знаннями – це сукупність стратегій та процесів щодо виявлення, набуття, розповсюдження, обміну та використання знань, необхідних для конкурентоспроможного бізнесу*	*Стратегія управління знаннями, процеси виявлення, формування, обміну та використання знань*
5.	*Е. Брукінг*	*Управління знаннями – це процес, який містить такі види діяльності: ідентифікація інтелектуального капіталу, документальне оформлення і створення бази знань, оновлення та захист знань, розповсюдження знань всередині організації*	*Оцінка інтелектуальних активів, ідентифікація знань за рівнями, база знань, примноження та захист знань, обмін знаннями*

Аналіз цих визначень дає змогу виявити таку проблему: знання цінуються як елемент управління бізнесом, а не як щоденна практика. Це пов’язано з відсутністю методик оцінки інтелектуальних активів та ефективністю їх використання, апробованих у сучасній українській практиці моделей створення інновацій та моделей організаційної культури знань.

Концепція створення системи управління знаннями відрізняється від традиційної системи управління підприємством. Критерієм ефективності традиційної системи управління вважається розмір прибутку, отриманий у результаті діяльності підприємства чи організації.

Предметом управління у системі управління знаннями вважається формування та використання різного рівня знань, які вимагають спеціальних підходів до стратегії та структури організації, створення нової організаційної культури та персональної політики. Ефективне управління знаннями, яке забезпечує конкурентні переваги, будується на поєднанні культури знань, процесу

отримання, розповсюдження та вдосконалення знань, а також використання нових інформаційних технологій. Отже, у цьому підході акценти зроблені на виробництво та розподіл знань всередині організації, створення організаційної культури з новими стратегічними цінностями.

В іншому підході увага концентрується на елементах інтелектуальних активів, класифікації знань, їх документуванні та юридичному захисті, ефективному використанні активів та концепції спірального нарощування знань.

У загальному випадку виділяють два протилежних підходи до управління знаннями:

Персоніфікуючий (інтуїтивістський) підхід – виходить з того, що носіями знань є люди, і головним акцентом в управлінні знаннями є співробітники підприємства, їх мотивація, зв'язки, культура компанії, тоді як технології є просто інфраструктурою. Прихильники цього підходу вважають, що при його використанні значно більше шансів уловити неформалізовані (такі, що не підлягають кодифікації) знання співробітників. Підхід має на увазі безпосереднє спілкування співробітників і передачу знань на зборах і тренінгах, а також створення умов для обміну досвідом. Таким чином, основне завдання управління знаннями відповідно до персоніфікуючого підходу – виявлення, збереження і ефективне використання знань співробітників.

Інформаційний (технологічний) підхід припускає, що ключовим чинником успіху є використання в процесі управління знаннями інформаційних технологій, які забезпечують виявлення прихованих залежностей і правил у накопичених підприємством даних і інформації та їх трансформації в масив знань.

Водночас очевидно, що необхідною є інтеграція запропонованих підходів і врахування всіх граней управління знаннями. Скористаємося концептуальною моделлю управління знаннями, запропонованою Симусом Галахером і Ширлі-Енн Хазлет, які виділяють три взаємопов'язані елементи: *інфраструктура знань, культура знань, технологія знань*. Згідно з цією моделлю, оскільки керівництво будь-якого підприємства, що прагне до ефективного управління знаннями, повинне, передусім, орієнтуватися на

існуючу структуру підприємства і процеси, а також знання, необхідні для виконання цих процесів, можна виділити *дві інфраструктури знань* (ІЗ). *Перша* включає існуючу структуру і процеси організації, *друга* відображає джерела знань. Загальна структура і бізнес-процеси визначатимуть організацію процесів управління знаннями, розподіл прав і обов'язків у системі управління. Друга складова інфраструктури знань – знання джерел знань або наявність мета-знання – визначається ідентифікованістю знань і досягається розробкою карт знань. Карти знань допомагають отримувати необхідні знання, вони визначають те знання, що є найбільш важливим для успіху організації, і вказують місце, де його можна отримати.

Культура знань (КЗ) передбачає усвідомлення важливості ролі людини як носія знань, і створення в людині зацікавленості в обміні знаннями. Підприємства, що займаються організацією управління знаннями, повинні забезпечити культурне середовище, яке б сприяло обміну знаннями і усуненню існуючих культурних бар'єрів.

До *основних технологій*, що підтримують управління знаннями (ТЗ), відносять:

1) дослідження даних і текстів (Data mining, Text Mining), що розуміються як розпізнавання образів, виділення значущих закономірностей з даних, що знаходяться в сховищах або вхідних чи вихідних потоках інформації;

2) системи управління документообігом (Document management), що забезпечують зберігання, архівацію, індексування, розмітку і публікацію документів;

3) засоби для організації спільної роботи (Collaboration) – мережі Intranet, технології групової роботи, синхронні й асинхронні конференції;

4) корпоративні портали знань;

5) засоби, що підтримують ухвалення рішень (Decision support) – експертні системи, системи підтримки прийняття рішень, дискусійні групи тощо.

Компоненти інформаційне забезпечення управління знаннями передбачають структуризацію за такими ознаками: *домени*

знань; джерела знань; типи знань; засоби виявлення нових знань; засоби оновлення знань.

Забезпеченість підприємства достовірною і достатньою інформацією по кожному з цих напрямів є ключовим чинником успіху в процесі управління знаннями.

Виходячи зі сказаного можна запропонувати таку модель управління знаннями підприємства, що враховує сучасну специфіку функціонування вітчизняних підприємств, яка представлена на рис. 1.1.

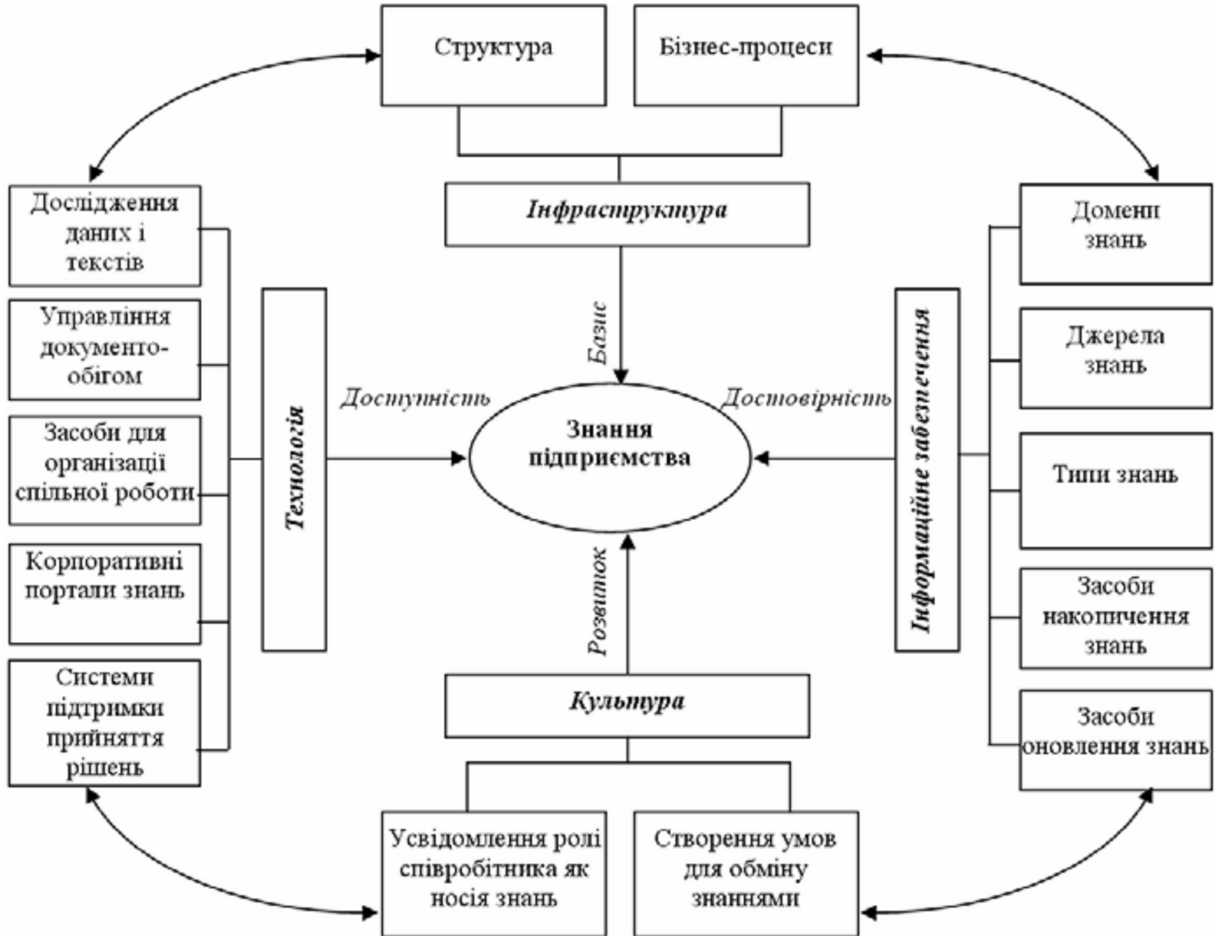

Рис. 1.1. Модель управління знаннями підприємства

Залежно від встановленої системи пріоритетів і поточного стану підприємства керівництво може приділяти більш пильну увагу певному аспекту, проте тільки за умови дотримання комплексного підходу процес управління знаннями підприємства може виявитися ефективним.

7. Поняття про моделі представлення знань в інтелектуальних системах

Центральним питанням побудови систем, заснованих на знаннях, є вибір форми представлення знань. Представлення знань – це спосіб формального вираження знань про предметну область у комп'ютерно-інтерпретованій формі. Відповідні формалізми, що забезпечують зазначене представлення, називають *моделями представлення знань.*

Модель представлення знань повинна відображати істотні характеристики задачі, що розв'язується, і забезпечувати відповідною інформацією процедури, що виконують пошук рішень. При цьому вона повинна володіти необхідною виразною здатністю, щоб відбивати всі важливі деталі предметної області задачі, а також бути досить ефективною з точки зору пошуку рішень, що розглядають як вивід на знаннях. Виразна сила й обчислювальна ефективність – дві основні характеристики моделей представлення знань. Багато моделей представлення знань, що володіють великими виразними можливостями, не піддаються ефективній реалізації. Тому пошук оптимального співвідношення між виразною силою моделі представлення знань, що використовується, і ефективністю її реалізації – основна задача розроблювача системи штучного інтелекту.

На сьогодні розроблено вже достатню кількість моделей. Кожна з них володіє своїми плюсами і мінусами, і тому для кожного конкретного завдання необхідно вибрати саме свою модель. Від цього буде залежати не стільки ефективність виконання поставленого завдання, скільки можливість її рішення взагалі.

Моделі представлення знань можна умовно розділити на *декларативні* і *процедурні*. У декларативних моделях знання представляються у вигляді описів об'єктів і відносин між об'єктами без вказівки в явному вигляді, як ці знання обробляти. Такі моделі припускають відділення описів (декларацій) інформаційних структур від механізму виведення, що оперує цими структурами.

У процедурних моделях значення представляються алгоритмами (процедурами), що містять необхідний опис інформаційних елементів і одночасно визначають способи їх обробки.

Конкретні моделі, застосовувані на практиці, являють собою комбінацію декларативних і процедурних представлень. Найбільш розповсюдженими є такі моделі представлення знань:

− логічні моделі;

− продукційні моделі;

− мережні моделі;

− фреймові моделі.

Ці моделі представлення знань будуть докладно розглянуті у наступній темі.

Питання для самоконтролю

1. Які визначення існують для понять інформація, дані та знання в інтелектуальних системах?
2. Які особливості мають знання порівняно з даними?
3. Які типи знань виокремлюють в інтелектуальних системах?
4. У чому полягає важливість управління знаннями і які можливості надає організації система управління знаннями?
5. Яка відмінність існує між концепцією створення системи управління знаннями та традиційною системою управління підприємством?
6. Що являє собою управління знаннями організації з практичного погляду?
7. Які існують підходи до управління знаннями?
8. У чому полягає мета системи управління знаннями?
9. Що розуміють під терміном «представлення знань»?
10. Які моделі представлення знань існують в інтелектуальних системах?

ТЕМА 2
МОДЕЛІ ПОДАННЯ ЗНАНЬ

1. Методи подання знань.
2. Підходи до обробки знань.
3. Найбільш поширені моделі подання знань.

1. Методи подання знань

Однією з найважливіших проблем, характерних для систем ШІ, є подання знань. Це пояснюється тим, що форма подання знань впливає на характеристики і властивості інформаційної системи (ІС). Проблема подання знань – це проблема подання взаємозв'язків у конкретній ПрО у формі, зрозумілій комп'ютеру.

Подання знань – домовленість про те, як описувати реальний світ. Вона потребує формалізації і структурування знань.

На відміну від методів подання даних, що базуються на певних алгоритмах, моделі подання знань використовують інформацію, що надходить від експерта. Ця інформація часто буває суперечливою, неповною, нечіткою, тому потрібна її формалізація, яка здійснюється за допомогою різних методів багатозначної логіки, теорії нечітких множин тощо.

2. Підходи до обробки знань

Існує два найпоширеніших підходи до обробки знань:

1. *Логічний* (формальний) підхід, при якому основна увага приділяється вивченню та застосуванню теоретичних методів подання знань, формалізації, а також логічній повноті (приміром, створення моделей подання знань на основі деяких логічних числень).

2. *Евристичний* (когнітивний) підхід, що орієнтується на забезпечення можливостей вирішення задач. На відміну від формальних евристичні моделі мають набір різноманітних засобів, що

передають специфічні особливості ПрО. Саме тому евристичні моделі перевершують логічні за виразністю і можливістю адекватно представляти ПрО.

3. Найбільш поширені моделі подання знань:

– формальні логічні моделі;
– продукційні моделі;
– семантичні мережі;
– фреймові моделі.

Логічні моделі

Основна ідея підходу при побудові *логічних моделей* подання знань полягає в тому, що вся інформація, необхідна для вирішення прикладних задач, розглядається як сукупність фактів і тверджень, які являють собою формули логіки. Знання відображаються сукупністю таких формул, а отримання нових знань зводиться до реалізації процедур логічного виведення.

В основі логічних моделей подання знань лежить поняття формальної теорії, що задається четвіркою:

S = <B, F, A, R>, де

B – множина базових символів (алфавіт);

F – множина формул;

A – множина аксіом, $A \subseteq F$;

R – скінченна множина відношень між формулами, що називається правилами виведення.

Переваги логічних моделей подання знань:

1. За фундамент тут використовується класичний апарат математичної логіки, методи якого досить добре вивчені і формально обґрунтовані.

2. Існують досить ефективні процедури виведення.

3. У БЗ можна зберігати лише множину аксіом, а решту знань отримувати з них за правилами виведення.

Продукційні моделі

Продукційні правила – це найпростіший засіб подання знань. Він базується на правилах, структурованих відповідно зраз-

ку «ЯКЩО *A*, ТОДІ *B*». *A* називається посиланням, а *B* – висновком або дією. Це означає, що «якщо всі умови *A* є істинними, то *B* також істинне» або «якщо всі умови *A* виконуються, то потрібно виконати дію *B*». Подібні правила зустрічаються в різноманітних галузях знань та видах діяльності.

Дослідження процесів прийняття рішення людиною свідчать, що людина в процесі міркування та пошуку рішень використовує саме продукційні правила.

Змінні, що містяться в *A* та *B*, показують, що правило містить якесь універсальне, загальне знання, абстраговане від конкретних значень цих змінних. Та сама змінна, що використана у логічному виведенні і різноманітних посиланнях, може приймати різноманітні конкретні значення.

Простота і наочність подання знань за допомогою продукцій обумовила його застосування в багатьох системах, що називаються продукційними.

Продукційне правило можна подати у такій формі:

$$i : S; L; A \to B; Q,$$

де *i* – індивідуальний номер продукції;

S – опис класу ситуацій, в яких ця продукція може використовуватися;

$A \to B$ – ядро продукції;

Q – постумова продукційного правила, що описує дії, які необхідно здійснити після виконання *B*.

Властивості продукційних моделей:

Модульність – окремі продукційні правила можуть бути додані, видалені або змінені в БЗ незалежно від інших.

Автономність – кожне продукційне правило – самостійний елемент знань (локальне джерело знань), окремі продукційні правила пов’язані одне з одним лише через потік даних, які вони обробляють.

Простота інтерпретації – «прозора» структура продукційних правил полегшує їхню змістовну інтерпретацію.

Природність – знання у вигляді «що робити і коли» є природними з погляду здорового глузду.

Недоліки продукційних систем виявляються тоді, коли значно зростає кількість правил.

Продукційні моделі є дуже поширеними засобами подання знань. Вони схожі на логічні моделі, тому на їх основі можна організовувати ефективні процедури виведення, але вони наочніше (порівняно з класичними логічними моделями) відображають знання. Продукційні моделі найчастіше застосовуються в промислових експертних системах.

Семантичні мережі

Семантичні мережі спроможні відображати структуру знань в усій складності їх взаємозв'язків, пов'язувати в єдине ціле об'єкти і їх властивості. Вони являють собою потужніші засоби подання знань порівняно з продукційними моделями.

Мережні моделі є найадекватнішим способом формалізації подання знань у природномовних текстах. Знання описуються сукупностями трійок *(arb)*, де *a* і *b* – об'єкти або поняття, а *r* – бінарне відношення між ними.

Формально мережні моделі подання знань можуть бути задані у вигляді $H = \langle I, C_1, ..., C_n, R \rangle$,

де *I* – множина інформаційних одиниць;

$C_1, ..., C_n, i = \overline{1, n}$ – множина типів зв'язків між елементами *I*;

R – відображення між інформаційними одиницями, що входять до *I*, з заданим набором типів зв'язків $\{C_i\}$.

Залежно від типів зв'язків $\{C_i\}$ розрізняють:

Мережі класифікації, в яких використовуються відношення структуризації, що дозволяють вводити в бази знань різні ієрархічні відношення між елементами множини *I*.

Функціональні мережі, в яких використовуються функціональні відношення, що дозволяють описувати процедури обчислень одних інформаційних одиниць через інші.

Сценарії, у яких використовуються казуальні відношення (приміром, причинно-наслідкові), а також відношення типів «засіб-результат», «знаряддя-дія» тощо.

Якщо в мережній моделі припускаються зв'язки різних типів, то її називають *семантичною мережею*. Термін «семантична» означає «змістовна», а «*семантика*» – встановлення взаємовідношень між символами й об'єктами, які вони позначають.

Семантична мережа – це мережна модель, яка формалізує знання у вигляді орієнтованого графа з розміченими вершинами і дугами. Вершинам відповідають об'єкти, поняття або ситуації, а дугам – відношення між ними. Це найбільш загальна модель подання знань, тому що в ній є засоби реалізації всіх характерних для знань властивостей: внутрішньої інтерпретації, структурованості, семантичної метрики й активності.

Переваги мережних моделей:

– великі виразні можливості;

– наочність системи знань, представленої графічно;

– близькість структури мережі, що являє собою систему знань, до семантичної структури фраз природною мовою;

– відповідність сучасним уявленням про організацію довготривалої пам'яті людини.

Недоліки мережних моделей:

– мережна модель не дає чіткого уявлення про структуру ПрО, що їй відповідає, тому формування і модифікація такої моделі є важкими;

– мережні моделі являють собою пасивні структури, для обробки яких необхідний спеціальний апарат формального висновку і планування;

– складність виведення на семантичних мережах, особливо за наявності багатьох відношень між її елементами, до якої призводить те, що пошук рішення у семантичній мережі зводиться до проблеми пошуку фрагмента мережі, що відповідає мережі поставленої задачі.

Фреймові моделі

Недоліки семантичних мереж спричинили появу синтагматичних ланцюгів, сценаріїв, фреймів тощо.

Термін «фрейм» (frame – рамка) було запропоновано в 1979 р. М. Мінським для визначення структури знань. Фрейм – це одиниця подання знань про об'єкт, яку можна описати певною сукупністю понять і сутностей. Внутрішня структура фрейму складається з множини елементів, що звуться слотами. Кожний слот – це певна структура даних або процедура. Фрейм – це абстрактне уявлення для подання деякого стереотипу сприйняття.

Формально фрейм має таку структуру:

$$f\left[\langle v_1, g_1\rangle, \langle v_2, g_2\rangle, ..., \langle v_k, g_k\rangle\right],$$

де f – ім'я фрейму;

пара $\langle v_i, g_i\rangle, i = \overline{1,k}$, – і-й слот, де v_i – ім'я слоту, g_i – його значення.

Фрейми поділяють на групи:

- *фрейми-структури*, що використовуються для визначення об'єктів та понять;
- рольові фрейми;
- фрейми-сценарії;
- фрейми-ситуації.

У рольовому фреймі імена слотів – це питання, відповіді на які є значеннями слотів. Якщо в загальному виразі для фрейму залишити тільки імена, то отримаємо конструкцію, яка у різних джерелах називається прототипом фрейму, просто фреймом або фреймом-інтенсіоналом. Фрейми з конкретними значеннями слотів називаються фреймами-прикладами, фреймами – екземплярами.

Фрейми мають властивість вкладеності, тобто як значення слоту може виступати система імен слотів глибшого рівня. Це забезпечує фреймовим мовам структурованість і зв'язність знань. Наявність імен фреймів і імен слотів означає, що знання, збережені у фреймах, мають характер посилань і тому вони внутрішньо інтерпретовані.

Основним недоліком фреймових моделей є відсутність механізмів керування виведенням.

Питання для самоконтролю

1. Чим відрізняються методи подання знань від моделей подання знань?
2. Які особливості має логічний підхід до обробки знань?
3. Які особливості має евристичний підхід до обробки знань?
4. У чому основна ідея підходу при побудові логічних моделей подання знань?
5. У чому основна ідея підходу при побудові продукційних моделей подання знань?
6. У чому основна ідея підходу при побудові семантичних моделей подання знань?
7. У чому основна ідея підходу при побудові фреймових моделей подання знань?

ТЕМА 3
ВИКОРИСТАННЯ НЕЧІТКОЇ ЛОГІКИ У СИСТЕМАХ, ЩО БАЗУЮТЬСЯ НА ЗНАННЯХ

1. Поняття про нечітку логіку та її місце у сучасних інформаційних системах.
2. Нечіткі множини.
3. Нечіткі змінні.
4. Операції нечіткої логіки.
5. Знання як нечітка система: функції та структура.
6. Фазифікація входів.
7. Нечітке логічне виведення.
8. Дефазифікація виходів.
9. Приклад нечіткого виведення.
10. Нечітка логіка в Matlab.

1. Поняття про нечітку логіку та її місце у сучасних інформаційних системах

Прийняття рішень у проблемно-орієнтованих інформаційних системах та системах управління здійснюється в умовах апріорної невизначеності, обумовленої неточністю або неповнотою вхідних даних, стохастичною природою зовнішніх впливів, відсутністю адекватної математичної моделі функціонування, нечіткістю мети, людським фактором та ін. Невизначеність системи призводить до зростання ризиків від прийняття неефективних рішень, результатом чого можуть бути негативні економічні, технічні та соціальні наслідки. Невизначеності у системах прийняття рішень компенсують за допомогою різноманітних методів штучного інтелекту. Для ефективного прийняття рішень при невизначеності умов функціонування системи застосовують методи на основі правил нечіткої логіки. Такі методи ґрунтуються на нечітких множинах і використовують лінгвістичні величини і висловлювання для опису стратегій прийняття рішень. Методи нечітких множин особливо корисні за відсутності точної математичної моделі функціонування системи. Теорія нечітких множин дає можливість застосувати для

прийняття рішень неточні та суб'єктивні експертні знання про предметну область без формалізації їх у вигляді традиційних математичних моделей.

Предметом нечіткої логіки є побудова моделей наближених міркувань людини і їх використання в комп'ютерних системах. Основи нечіткої логіки були закладені наприкінці 60-х років у працях відомого американського математика Латфі Заде. Для створення дійсно інтелектуальних систем, здатних адекватно взаємодіяти з людиною, необхідний був новий математичний апарат, що переводить невиразні і неоднозначні життєві твердження в мову чітких і формальних математичних формул.

Апарат теорії нечітких множин, продемонструвавши ряд багатообіцяючих можливостей застосування – від систем керування літальними апаратами до прогнозування підсумків виборів, виявився разом з тим надмірно складним для втілення, враховуючи наявний на той час рівень технології, і на багато років нечітка логіка зайняла своє місце в ряді інших спеціальних наукових дисциплін – десь посередині між експертними системами і нейронними мережами.

Своє друге народження теорія нечіткої логіки пережила на початку вісімдесятих років минулого століття, коли відразу кілька груп дослідників (в основному в США і Японії) всерйоз зайнялися створенням електронних систем різного застосування, що використовують нечіткі керуючі алгоритми. Теоретичні основи для цих спроб були закладені в ранніх працях Коско й інших учених.

Третій період почався з кінця 80-х років і триває дотепер. Цей період характеризується бумом практичного застосування теорії нечіткої логіки в різних сферах науки і техніки. До 90-го року минулого століття з'явилося близько 40 патентів, що відносились до нечіткої логіки (30 японських). Сорок вісім японських компаній утворили спільну лабораторію LIFE (Laboratory for International Fuzzy Engineering), японський уряд фінансував 5-річну програму з нечіткої логіки, що включає 19 різних проектів – від систем оцінки глобального забруднення атмосфери і передбачення землетрусів до АСУ заводських цехів і складів. Результатом виконання цієї програми з'явилася поява цілого ряду нових

масових мікрочіпів, заснованих на нечіткій логіці. Сьогодні їх можна знайти в пральних машинах і відеокамерах, цехах заводів і моторних відсіків автомобілів, у системах керування складськими роботами і бойовими вертольотами.

У США розвиток нечіткої логіки йде шляхом створення систем, що потрібні великому бізнесу і військовим. Нечітка логіка застосовується при аналізі нових ринків, біржовій грі, оцінці політичних рейтингів, виборі оптимальної цінової стратегії та ін. З використанням теорії нечітких множин вирішуються питання узгодження суперечливих критеріїв прийняття рішень, створення логічних регуляторів систем. Нечіткі множини дають змогу застосовувати лінгвістичний опис складних процесів, встановлювати нечіткі відношення між поняттями, прогнозувати поведінку системи, формувати множину альтернативних дій, виконувати формальний опис нечітких правил прийняття рішень.

Перспектива застосування нечіткої логіки полягає у розробленні гібридних методів штучного інтелекту, до яких можна віднести нечіткі штучні нейронні мережі, адаптивне поповнення баз нечітких правил, підтримка нечітких запитів до баз даних, побудова нечітких когнітивних карт, нечіткі графи, нечіткі мережі Петрі, нечіткі дерева прийняття рішень, нечітка кластеризація та ін.

2. Нечіткі множини

Нехай $X \subseteq R^1$ – простір значень вхідних параметрів системи. Тоді нечітка або розмита множина (fuzzy set) *A* визначається на носії *X* у вигляді сукупності впорядкованих пар $(x, \mu_A(x))$:

$$A = \{x, \mu_A(x) \,|\, x \in X, 0 \le \mu_A(x) \le 1\},$$

де $\mu_A(x)$ – функція приналежності кожного *x* множині *A*.

Для дискретного носія нечітка множина позначається у вигляді:

$$A = \{x_1 / \mu_A(x_1),\ x_2 / \mu_A(x_2), \ldots, x_m / \mu_A(x_m)\}.$$

Функція приналежності $\mu_A(x)$ ставить у відповідність кожному $x \in X$ дійсне число з відрізку [0,1]. Найбільшого поширення отримали гаусова, сигмоідальна, поліноміальна, трикутна та

трапецієподібна функції приналежності. Конкретний вигляд функції визначається потребами досліджуваної предметної області.

Загальна форма трапецієподібної функції приналежності задається у вигляді:

$$\mu_A(x) = \begin{cases} \max(0,(x_4 - x)/(x_4 - x_3), & x > x_3 \\ \max(0,(x - x_1)/(x_2 - x_1), & x < x_2 \\ 1, & x \geq x_2 \; and \; x \leq x_3 \end{cases},$$

де $x_1 < x_2 < x_3 < x_4$. Частковим випадком трапецієподібної функції приналежності є трикутна функція приналежності, яка набуває значення 1 тільки в одній точці і визначає нечітке число.

3. Нечіткі змінні

У прикладних дослідженнях з проблем управління широко використовуються експертні оцінки, які формулюються у термінах природної мови. З цією метою використовуються нечіткі лінгвістичні змінні.

Нечітка лінгвістична змінна $\tilde{X}$ задається набором термів, які позначають якісні ознаки станів системи (наприклад, «низький», «середній», «високий»). Нехай $\tilde{X}$ визначається *m* лінгвістичними термами:

$$\tilde{X} = \{A_j \mid j = 1..m\}.$$

Тоді лінгвістична змінна $\tilde{X}$ є нечітким образом носія X. Кожен із термів є нечіткою множиною:

$$A_j = \{x, \mu_{A_j}(x) \mid x \in X, 0 \leq \mu_{A_j}(x) \leq 1\}.$$

Функції приналежності однієї лінгвістичної змінної $\tilde{X}$:

$$\mu_{A_j}(x), \quad j = 1..m$$

визначаються в одному вимірювальному просторі X.

4. Операції нечіткої логіки

Над нечіткими множинами можна виконувати логічні операції, аналогічні операціям звичайної (однозначної) логіки, наприклад, *NOT*, *AND*, *OR*. Множиною значень функції істинності звичайної логіки є {0,1}. Функції істинності нечіткої логіки набувають значення на відрізку [0,1] . У нечіткій логіці формула

(висловлювання) може бути істинною зі значенням $\mu \in [0,1]$. Нехай *A*, *B* – дві нечіткі множини.

4.1. Операція NOT – нечітке доповнення множини:

$$NOT\ A = A' = \{x, \mu_{A'}(x) \mid x \in X, \mu_{A'}(x) = 1 - \mu_A(x)\}.$$

Для нечітких величин операція *NOT* (¬) є *інволюцією* на [0,1]:

$$\neg\, a : [0,1] \to [1,0].$$

4.2. Операція AND ($\cap$) – нечітка кон'юнкція множин:

$$A\ AND\ B = A \cap B = T_{\cap}(\mu_A(x), \mu_B(x)).$$

Оператор $T_{\cap}$ є будь-яким бінарним відношенням, яке задовольняє умови:

1) обмеження

$$T_{\cap}(0,0) = 0, T_{\cap}(\mu,1) = T_{\cap}(1,\mu) = \mu;$$

2) монотонність

$$if\ \mu_A \le \mu_C\ and\ \mu_B \le \mu_D\ then\ T_{\cap}(\mu_A, \mu_B) \le T_{\cap}(\mu_C, \mu_D);$$

3) комутативність

$$T_{\cap}(\mu_A, \mu_B) = T_{\cap}(\mu_B, \mu_A);$$

4) асоціативність

$$T_{\cap}(T_{\cap}(\mu_A, \mu_B), \mu_C) = T_{\cap}(\mu_A, T_{\cap}(\mu_B, \mu_C)).$$

Операція *AND* ($\wedge$) над нечіткими величинами є триангулярною нормою і набуває значення на відрізку [0,1]:

$$a \wedge b : [0,1] \times [0,1] \to [0,1].$$

Одним із прикладів *t*-норми є знаходження мінімуму двох функцій приналежності:

$$A\ AND\ B = A \cap B = \min_x (\mu_A(x), \mu_B(x)).$$

4.3. Операція OR ($\cup$) – нечітка диз'юнкція множин:

$$A\ OR\ B = A \cup B = T_{\cup}(\mu_A(x), \mu_B(x)).$$

Аналогічно $T_{\cap}$, оператор $T_{\cup}$ є будь-яким бінарним відношенням, яке задовольняє умови:

1) обмеження:

$$T_{\cup}(1,1) = 1, T_{\cup}(\mu,0) = T_{\cup}(0,\mu) = \mu;$$

2) монотонність:

$$if\ \mu_A \le \mu_C\ and\ \mu_B \le \mu_D\ then\ T_{\cup}(\mu_A, \mu_B) \le T_{\cup}(\mu_C, \mu_D);$$

3) комутативність:

$$T_{\cup}(\mu_A,\mu_B)=T_{\cup}(\mu_B,\mu_A);$$

4) асоціативність:

$$T_{\cup}(T_{\cup}(\mu_A,\mu_B),\mu_C)=T_{\cup}(\mu_A,T_{\cup}(\mu_B,\mu_C)).$$

Операція *OR* ($\dot{\vee}$) над нечіткими величинами є триангулярною конормою, яка набуває значення на відрізку [0,1] :

$$a \dot{\vee} b : [0,1]\times[0,1]\to[0,1].$$

Одним із прикладів *t*-конорми є знаходження максимуму двох функцій приналежності:

$$A\,OR\,B = A\cup B = \max_x(\mu_A(x),\mu_B(x)).$$

Операція імплікації $T_{\to}$ є будь-яким бінарним відношенням, яке задовольняє умови:

1) обмеження:

$$T_{\to}(\mu,1)=1, T_{\to}(1,\mu)=\mu, T_{\to}(0,\mu)=1;$$

2) зміна порядку за першим аргументом:

$$if\ \mu_A\le\mu_B\ then\ T_{\to}(\mu_A,\mu_C)\ge T_{\cup}(\mu_B,\mu_C);$$

3) збереження порядку за другим аргументом:

$$if\ \mu_C\le\mu_D\ then\ T_{\to}(\mu_A,\mu_C)\le T_{\cup}(\mu_A,\mu_D).$$

Одним із прикладів такої операції є імплікація Кліна–Дайнеса:

$$A\to B=\max_x(1-\mu_A(x),\mu_B(x)).$$

Імплікація ($\to$) над нечіткими величинами є триангулярною операцією зі значенням на відрізку [0,1] :

$$a\to b:[0,1]\times[0,1]\to[0,1].$$

Імплікація над нечіткими величинами *a*, *b* може бути отримана за допомогою операцій нечіткого доповнення, норми та конорми, наприклад:

$$a\to b=\bar{a}\wedge\bar{b},\ a\to b=\bar{a}\vee b.$$

5. Знання як нечітка система: функції та структура

Знання можна формалізувати у вигляді системи нечітких логічних висловлювань.

Нехай нечітка система здійснює вибір варіантів рішень на основі залежності вихідної величини від декількох вхідних величин. Припустимо, що математична модель залежності виходу від входів відсутня і замість неї використовується база експертних правил у вигляді нечітких висловлювань «*if -then*» у термінах лінгвістичних змінних та нечітких множин.

Тоді функціональність нечіткої системи прийняття рішень визначається такими кроками:

1) перетворення чітких вхідних змінних на нечіткі, тобто визначення ступеня відповідності входів кожній із нечітких множин;

2) обчислення правил на основі використання нечітких операторів та застосування імплікації для отримання вихідних значень правил;

3) агрегування нечітких виходів правил у загальне вихідне значення;

4) перетворення нечіткого виходу правил на чітке значення.

6. Фазифікація входів

Фазифікація полягає у перетворенні чітких вхідних величин:

$$\overline{x} = (x_1, x_2, \ldots, x_n)$$

на нечіткі множини:

$$A' = (A'_1, A'_2, \ldots, A'_n) .$$

У більшості випадків для цього використовуються *сингле-тонні моделі*.

Синглетон чіткого значення x_i є нечіткою множиною $A'_i(x, \mu_{A'_i}(x))$ з функцією приналежності:

$$\mu_{A'_i}(x) = \begin{cases} 1, x = x_i; \\ 0, x \neq x_i. \end{cases}$$

При фазифікації чіткого входу x_i визначають ступені його відповідності кожному лінгвістичному терму $A_{i,j}$ з функціями приналежності $\mu_{A_{i,j}}(x)$, $j = 1..m_i$. Ці ступені є значеннями функцій приналежності $\mu_{A_{i,j}}(x)$ у точці $x = x_i$, або інакше – значеннями $A_{i,j}(x_i)$, i =1..n .

7. Нечітке логічне виведення

Нечіткі вхідні значення системи перетворюються на вихідні на основі правил нечіткої логіки, що характерно для експертних систем прийняття рішень. Нехай система прийняття рішень здійснює перетворення значень *n* вхідних лінгвістичних змінних $\tilde{X} = \{\tilde{X}_i \mid i = 1..n\}$ у вихідну лінгвістичну змінну $\tilde{Y} = R(\tilde{X})$ згідно з базою правил $R = \{R_k \mid k = 1..N\}$. Правила *R* акумулюють знання експертів у вигляді нечіткої імплікації $R = A \to B$, яку можна розглядати як нечітку множину на декартовому добутку носіїв вхідних та вихідних розмитих множин. Процес отримання нечіткого результату B' з нечітких вхідних множин A' на основі знань $A \to B$ можна зобразити у такому вигляді:

$$B' = A' \bullet R = A' \bullet (A \to B),$$

де $\bullet$ – композиційне правило нечіткого виведення.

На практиці для нечіткого виведення використовується *максимінна композиція*, а нечітка імплікація реалізується знаходженням мінімуму функцій приналежності.

Для імітації роботи експертної системи за схемою імплікації використовується множина нечітких продукційних правил, кожне з яких будується у вигляді умовного оператора:

if *логічний вираз* ***then*** *оператор*,

де *логічний вираз* – висловлювання, побудоване на основі базових логічних операцій над нечіткими величинами; *оператор* – результуюче рішення. Правила можуть визначати відношення відповідності (is) між вхідними лінгвістичними змінними $\tilde{X}$ та їх нечіткими термами $\{A_{i,j} \mid i = 1,..,n; j = 1..m_i\}$. Використання нечітких умовних правил є природнім для подання знань експертами і спрощує їх машинне опрацювання.

Загалом до правила можуть входити всі можливі комбінації лінгвістичних термів для усіх вхідних змінних, об’єднаних логічними операціями. Слід зазначити, що за допомогою перетворень нечітких множин будь-яке правило, що містить у лівій частині як кон’юнкції, так і диз’юнкції, можна перетворити на систему правил, у лівій частині яких будуть або тільки кон’юнкції, або тільки

диз'юнкції. Для визначення нечіткої кон'юнкції можна використати знаходження мінімуму, а для нечіткої диз'юнкції – знаходження максимуму двох функцій приналежності. Не зменшуючи загальності, розглядатимемо правила, побудовані на основі кон'юнкції. Розрізняють дві моделі логічного виведення: Мамдані (Mamdani) та Сугено (Sugeno).

Модель Мамдані оперує лише лінгвістичними змінними та нечіткими множинами і перетворює нечіткі входи на нечіткі виходи. Наприклад, для моделі Мамдані правила мають вигляд:

$$R_k : if\ \tilde{X}_1\ is\ A_{1,k}\ and\ ...and\ \tilde{X}_n\ is\ A_{n,k}\ then\ \tilde{Y}\ is\ B_k\ ,$$

де $A_{i.k} \in \tilde{X}_i$ – нечіткі множини для вхідних та $B_k \in B$ – нечіткі множини для вихідної лінгвістичних змінних, які використовуються в k -му правилі (k □1..N). Операція *and* інтерпретується як *t*-норма нечітких множин.

Модель Сугено оперує з чіткими величинами, лінгвістичними змінними та нечіткими множинами і перетворює чіткі входи у чіткі виходи. Правила моделі Сугено можуть мати вигляд:

$$R_k : if\ x_1\ is\ A_{1,k}\ and\ ...and\ x_n\ is\ A_{1,n}\ then\ \ y = o_k\ ,$$

де o_k – завершальне значення k -го правила, вихідний сигнал або керуюча дія.

Для повноти бази нечітких правил повинні виконуватися такі умови:

1) для будь-якого терму вхідної змінної існує хоча б одне правило, в якому цей терм використовується у лівій частині правила;

2) існує хоча б одне правило для кожного лінгвістичного терму вихідної змінної.

Для багатовходових систем застосовується механізм логічного виведення, характерною рисою якого є використання рівнів істинності передумов правил. Для кожного правила R_k , k □ 1..N визначається рівень його істинності α_k щодо входів. Рівень істинності є дійсним числом, яке характеризує ступінь відповідності нечітких входів системи A_i' , $i = 1..n$ заданим у правилах нечітким множинам $A_{i,j}$ ($j = 1..m_i$):

$$\alpha_k = \min_{i=1}^{n}\left[\max_{X_i}\left(A_i' \wedge A_{i,j}\right)\right],$$

де X_i – простір визначення входів A_i', $i = 1..n$; операція $\wedge$ – нечітка кон'юнкція.

При використанні вхідних синглетонів механізм логічних виведень спрощується, оскільки ступінь істинності правил може бути визначений на основі фазифікованих входів:

$$\max_{X_i}\left(A_i'(x_i) \wedge A_{i,j}\right) = A_{i,j}(x_i).$$

У цьому випадку обчислення рівня істинності *k*-го правила формуватиметься за формулою:

$$\alpha_k = \min_i\left(A_{i,j}(x_i)\right).$$

Кожне із правил є нечіткою імплікацією, яка визначає вихідне значення залежно від рівня істинності лівої частини правила. Ступінь впевненості виведення задається функцією приналежності відповідного вихідного терму B_k. Використовуючи один зі способів побудови нечіткої імплікації, одержимо нові нечіткі змінні, або відповідні ступені впевненості в значенні виходів при застосуванні відповідного правила до заданих входів. Так, на основі визначення нечіткої імплікації за Мамдані, як мінімуму лівої й правої частин правила, маємо:

$$B_k' = \min(\alpha_k, B_k), \quad k = 1..N,$$

де B_k' – зрізи вихідних нечітких множин на рівні α_k.

Завершальним кроком нечіткого логічного виведення є агрегування виходів правил. Один з основних способів акумуляції – нечітка диз'юнкція вихідних множин, або, інакше, знаходження максимуму отриманих функцій приналежності. Як результат, отримаємо значення агрегованого виходу:

$$B' = \max_k\left(B_k'\right), \quad k = 1..N.$$

При нечіткому логічному виведенні паралельно опрацьовують велику кількість правил з подальшим їх агрегуванням у завершальне рішення. Правила можуть будуватися на основі досвіду та знань експертів, створенням моделі дій оператора, методом навчання. При проектуванні пристроїв з нечіткою логікою важливо забезпечити можливості їх пристосування до змін навколишнього

середовища методом навчання бази правил за експериментальними даними.

Навчання полягає в адаптивному підборі параметрів нечітких множин та автоматичному генеруванні правил нечіткого логічного виведення. Для цього використовуються алгоритми оптимізації та інтелектуального опрацювання даних – градієнтний, генетичний, штучних нейронних мереж, байесових мереж та ін.

8. Дефазифікація виходів

Після визначення індивідуальних виходів правил здійснюється *дефазифікація* агрегованого виходу. У загальному випадку етап дефазифікації є необов'язковим і використовується за необхідності перетворення виведених нечітких лінгвістичних змінних до точного значення.

Існує декілька методів дефазифікації – *метод середнього центру, перший максимум, середній максимум, висотна дефазифікація*.

Метод середнього центру, або центроїдний метод, полягає у визначенні центру ваги вихідної нечіткої множини:

$$y = \frac{\sum_{j=1}^{m} y_j B'(y_j)}{\sum_{j=1}^{m} B'(y_j)}.$$

Для моделі Сугено вихідні множини правил задаються у вигляді сінглетонів з функціями приналежності

$$\mu_{o_k}(y_k) = \begin{cases} 1, & y_k = o_k \\ 0, & y_k <> o_k \end{cases},$$

де o_k – вихідне значення k -го правила.

Тоді результуюче чітке вихідне значення системи прийняття рішень обчислюється зважуванням значень активованих правил:

$$y = \frac{\sum_{k=1}^{N} \alpha_k y_k}{\sum_{k=1}^{N} \alpha_k}.$$

У системах управління отримане чітке вихідне значення використовується у контурі зворотного зв’язку для вироблення керуючих дій.

9. Приклад нечіткого виведення

Нехай база нечітких правил прийняття рішень містить визначені експертами залежності шансу працевлаштування молодого спеціаліста від його рівня підготовки (рейтингу) та попиту на спеціалістів такого профілю на ринку праці.

Введемо лінгвістичні змінні:

- *рейтинг = (високий, низький);*
- *попит = (високий, низький);*
- *шанс = (високий, середній, низький).*

Наведемо декілька із можливих правил:

R_1: якщо *попит* є *високим* і *рейтинг* є *високим*, то *шанс* є *високим*;

R_2: якщо *попит* є *низьким* і *рейтинг* є *високим*, то *шанс* є *середнім*;

R_3: якщо *попит* є *високим* і *рейтинг* є *низьким*, то *шанс* є *середнім*.

Припустимо, що лінгвістичні терми входів описуються такими нечіткими множинами:

високий попит = {100/0.2; 200/0.4; 300/0.8; 400/1};

низький попит = {100/1; 200/0.8; 300/0.6; 400/0.4};

високий рейтинг = {50/0.1; 71/0.8; 88/0.9; 100/1};

низький рейтинг = {50/1; 71/0.3; 88/0.2; 100/0.1}.

Терми виходу описуються такими множинами:

високий шанс = {0/0.1; 0.5/0.5; 1/1};

середній шанс = {0/0.5; 0.5/1; 1/0.5}.

Необхідно визначити шанс працевлаштування при *незначному попиті* та *середньому рейтингу*.

Звернемо увагу на те, що вхідні дані не визначають термів *незначний попит* та *середній рейтинг*. Вихідну реакцію на ці нечіткі значення необхідно отримати в процесі логічного виведення на основі бази правил.

Нехай на вхід системи надходять нечіткі множини попиту

$$A_1' = \{100/0.4;\ 200/0.3;\ 300/0.2;\ 400/0.1\}$$

та рейтингу

$$A_2' = \{50/0.1;\ 71/0.8;\ 88/0.8;\ 100/0.1\}.$$

Операції визначення мінімуму та максимуму позначимо у вигляді $\wedge$ та $\vee$ відповідно.

Для обчислення виходу виконаємо етапи нечіткого логічного виведення:

1. Обчислення рівнів істинності правил:

$$\alpha_1 = \min[\max(0.4\text{^}0.2,\ 0.3\text{^}0.4,\ 0.2\text{^}0.8,\ 0.1\text{^}1),\ \max(0.1\text{^}0.1,\ 0.8\text{^}0.8,\ 0.8\text{^}0.9,\ 0.1\text{^}1)] =$$
$$= \min[\max(0.2,\ 0.3,\ 0.2,\ 0.1),\ \max(0.1,\ 0.8,\ 0.8,\ 0.1)] = \min[0.3,\ 0.8] = 0.3$$
$$\alpha_2 = \min[\max(0.4\text{^}1,\ 0.3\text{^}0.8,\ 0.2\text{^}0.6,\ 0.1\text{^}0.4),\ \max(0.1\text{^}0.1,\ 0.8\text{^}0.8,\ 0.8\text{^}0.9,\ 0.1\text{^}1)] =$$
$$= \min[\max(0.4,\ 0.3,\ 0.2,\ 0.1),\ \max(0.1,\ 0.8,\ 0.8,\ 0.1)] = \min[0.4,\ 0.8] = 0.4$$
$$\alpha_3 = \min[\max(0.4\text{^}0.2,\ 0.3\text{^}0.4,\ 0.2\text{^}0.8,\ 0.1\text{^}1),\ \max(0.1\text{^}1,\ 0.8\text{^}0.3,\ 0.8\text{^}0.2,\ 0.1\text{^}0.1)] =$$
$$= \min[\max(0.2,\ 0.3,\ 0.2,\ 0.1),\ \max(0.1,\ 0.3,\ 0.2,\ 0.1)] = \min[0.3,\ 0.3] = 0.3$$

2. Обчислення виходів правил:

$$B_1' = \{0/\min(0.3,\ 0.1),\ 0.5/\min(0.3,\ 0.5),\ 1/\min(0.3,\ 1)\} = \{0/0.1,\ 0.5/0.3,\ 1/0.3\}$$
$$B_2' = \{0/\min(0.4,\ 0.5),\ 0.5/\min(0.4,\ 1),\ 1/\min(0.4,\ 0.5)\} = \{0/0.4,\ 0.5/0.4,\ 1/0.4\}$$
$$B_3' = \{0/\min(0.3,\ 0.5),\ 0.5/\min(0.3,\ 1),\ 1/\min(0.3,\ 0.5)\} = \{0/0.3,\ 0.5/0.3,\ 1/0.3\}$$

3. Агрегування виходів:

$$B' = B_1' \vee B_2' \vee B_3' = \{0/\max(0.1,\ 0.4,\ 0.3),\ 0.5/\max(0.3,\ 0.4,\ 0.3),\ 1/\max(0.3,\ 0.4,\ 0.3)\} =$$
$$= \{0/0.4,\ 0.5/\ 0.4,\ 1/0.4\}$$

4. Дефазифікація виходу:

$$y = \frac{0*0.4 + 0.5*0.4 + 1*0.4}{0.4 + 0.4 + 0.4} = \frac{0.6}{1.2} = 0.50$$

Отже, для заданих нечітких множин при незначному попиті на спеціалістів та середньому кваліфікаційному рейтингу шанс працевлаштуватися становить 50 %.

10. Нечітка логіка в Matlab

Fuzzy logic toolbox – вбудована в Matlab сукупність функцій, що містить набір засобів, які дозволяють:

- створювати і редагувати нечіткі системи всередині середовища Matlab;
- вбудовувати нечітку підсистему в SimuLink (поставляється з Matlab) при моделюванні загальної системи;

- побудувати нечітку систему в Matlab у вигляді процедури, що викликається з програми, яка написана на мові Сі.

Цей набір інструментів забезпечує три категорії інструментальних засобів програмування нечітких систем:

- функції командного рядка (command line functions);
- графічний інтерактивний інтерфейс;
- використання вбудованих блоків SimuLink.

Перша категорія – готові функції, які можна викликати відразу з командного рядка Matlab. Практично всі вони являють собою м-файли, що містять послідовність виразів, які виконують спеціалізований нечіткий алгоритм. Для перегляду вихідного коду функцій необхідно набрати в командному рядку:

type ім'я_функції.

Крім того, Matlab дозволяє їх модифікувати шляхом копіювання і перейменування відповідного файлу та наступного його редагування. Таким чином, нечіткий набір інструментів є розширеним власними функціями.

Друга категорія дозволяє отримати доступ до тих самих функцій через графічний користувальницький інтерфейс, за допомогою якого набагато зручніше конструювати й аналізувати нечіткі системи.

Третя категорія – моделювання в середовищі SimuLink. Тут підсистеми представляються у виді блоків – можна з'єднати будь-яким чином і відразу отримати результати.

У Matlab є багато вбудованих функцій приналежності, зокрема:

- *сигмоїдальна;*
- *двостороння сигмоїдальна;*
- *гаусова;*
- *дзвоноподібної форми;*
- *S-функція приналежності;*
- *Z-функція приналежності;*
- *трапецієподібна;*
- *трикутна й ін.*

Усі дії над нечіткими числами задаються мінімальним набором функцій і відбуваються всередині програми. Таким чином,

користувачу необов’язково вивчати всі тонкощі теорії нечітких множин, достатньо лише визначити всі вхідні і вихідні змінні і задати таблицю правил, а решту роботи робить Matlab. Дефазіфікація виконується в один з п’ятьох методів, зазначених програмістом. Крім того, можна вивести на екран відповідно до введених правил результуючі поверхні керування, залежно від комбінації входів, схему отриманої нечіткої програми, і це лише мала частина всіх можливостей даного набору інструментів.

FIS-редактор призначений для створення, збереження, завантаження і виведення на друк систем нечіткого логічного виведення, а також для редагування наступних властивостей: тип системи; найменування системи; кількість вхідних і вихідних змінних; найменування вхідних і вихідних змінних; параметри нечіткого логічного виведення.

Завантаження FIS-редактора відбувається за допомогою команди **fuzzy**. У результаті з’являється інтерактивне графічне вікно (рис. 3.1). У нижній частині графічного вікна FIS-редактори розташовані кнопки **Help** і **Close**, що дозволяють викликати вікно довідки і закрити редактор відповідно.

FIS-редактор містить 8 меню. Це три загальносистемних меню – **File**, **Edit**, **View** і п’ять меню для вибору параметрів нечіткого логічного виведення – **And Method**, **Or Method**, **Implication**, **Aggregation** і **Defuzzification**.

Меню File – це загальне меню для всіх GUI-модулів, використовуваних із системами нечіткого логічного виведення.

За допомогою команди **New FIS...** користувач має можливість створити нову систему нечіткого логічного виведення. При виборі цієї команди з’являться дві альтернативи: **Mamdani** і **Sugeno**, що визначають тип створюваної системи. Створити систему типу Mamdani можна також натисканням Ctrl+N.

За допомогою команди **Import** користувач має можливість завантажити раніше створену систему нечіткого логічного виведення. При виборі цієї команди з’являться дві альтернативи **From Workspace...** і **From disk**, що дозволяють завантажити систему нечіткого логічного виведення з робочої області MatLab і з диска відповідно. При виборі команди **From Workspace...** з’явиться

діалогове вікно, у якому необхідно вказати ідентифікатор системи нечіткого логічного виведення, що знаходиться в робочій області MatLab. При виборі команди **From disk** з'явиться діалогове вікно, у якому необхідно вказати ім'я файлу системи нечіткого логічного виведення. Файли систем нечіткого логічного виведення мають розширення .fis . Завантажити систему нечіткого логічного виведення з диска можна також натисканням Ctrl+N чи командою **fuzzy FIS_name**, де FIS_name – ім'я файлу системи нечіткого логічного виведення.

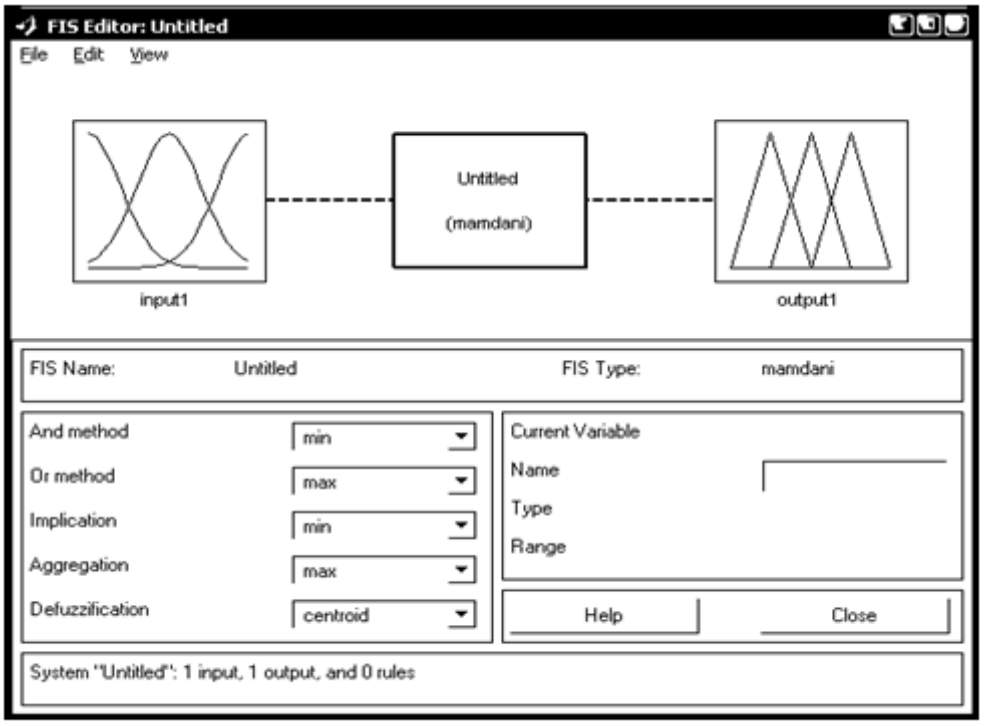

Рис. 3.1. Вікно FIS-редактора

При виборі команди **Export** з'являться дві альтернативи **To Workspace...** і **To disk**, що дозволяють скопіювати систему нечіткого логічного виведення в робочу область MatLab і на диск відповідно. При виборі команди **To Workspace...** з'явиться діалогове вікно, у якому необхідно вказати ідентифікатор системи нечіткого логічного виведення, під яким вона буде збережена в робочій області MatLab. При виборі команди **To disk** з'явиться діалогове вікно, у якому необхідно вказати ім'я файлу системи нечіткого логічного виведення. Скопіювати систему нечіткого логічного

виведення в робочу область і на диск можна також натисканням Ctrl+T і Ctrl+S відповідно.

Команда **Print** дозволяє вивести на принтер копію графічного вікна. Друк можливий також при натисканні Ctrl+P.

Команда **Close** закриває графічне вікно. Закриття графічного вікна відбувається при натисканні Ctrl+W чи однократному клацанні лівою кнопкою миші на кнопці Close.

Меню Edit: Команда **Undo** скасовує раніше зроблену дію. Виконується також по натисканню Ctrl+Z. Команда **Add Variable...** дозволяє додати в систему нечіткого логічного виведення ще одну змінну. При виборі цієї команди з'являться дві альтернативи **Input** і**Output**, що дозволяють додати вхідну і вихідну змінну відповідно. Команда **Remove Selected Variable** видаляє поточну змінну із системи. Ознакою поточної змінної є червона окантовка її прямокутника. Призначення поточної змінної відбувається за допомогою однократного клика лівої кнопки миші по її прямокутнику. Видалити поточну змінну можна також за допомогою натискання Ctrl+X.

Команда **Membership Function...** відкриває редактор функцій приналежності. Ця команда може бути також виконана натисканням Ctrl+2.

Команда **Rules…** відкриває *редактор бази знань.* Ця команда може бути також виконана натисканням Ctrl+3.

Меню View – це загальне меню для всіх GUI-модулів, використовуваних із системами нечіткого логічного виведення. Це меню дозволяє відкрити вікно візуалізації нечіткого логічного виведення (команда Rules або натискання клавіш Ctrl+5) і вікно виведення поверхні «вхід-вихід», що відповідає системі нечіткого логічного виведення (команда Surface або натискання клавіш Ctrl+6).

Меню And Method – це меню дозволяє установити наступні реалізації логічної операції "ТА": min – мінімум; prod – множення.

Користувач також має можливість установити власну реалізацію операції "ТА". Для цього необхідно вибрати команду **Custom…** і в графічному вікні, що з'явилося, надрукувати ім'я функції, що реалізує цю операцію.

Меню Or Method – це меню дозволяє установити такі реалізації логічної операції "АБО": **max** – множення; **probor** – імовірнісне "АБО".

Користувач також має можливість установити власну реалізацію операції "АБО". Для цього необхідно вибрати команду **Custom…** і в графічному вікні, що з'явилося, надрукувати ім'я функції, що реалізує цю операцію.

Меню Implication – це меню дозволяє установити такі реалізації імплікації: **min** – мінімум; **prod** – множення.

Користувач також має можливість установити власну реалізацію імплікації. Для цього необхідно вибрати команду **Custom…** і в графічному вікні, що з'явилося, надрукувати ім'я функції, що реалізує цю операцію.

Меню Aggregation – це меню дозволяє установити наступні реалізації операції об'єднання функцій приналежності вихідної змінної: **max** – максимум; **sum** – сума; **probor** – імовірнісне "АБО".

Користувач також має можливість установити власну реалізацію цієї операції. Для цього необхідно вибрати команду **Custom…** і в графічному вікні, що з'явилося, надрукувати ім'я функції, що реалізує цю операцію.

Меню Defuzzification – це меню дозволяє вибрати метод дефазифікації. Для систем типу Мамдані запрограмовані такі методи: **centroid** – центр ваги; **bisector** – медіана; **lom** – найбільший з максимумів; **som** – найменший з максимумів; **mom** – середнє з максимумів. Для систем типу Сугено запрограмовані такі методи: **wtaver** – зважене середнє; **wtsum** – зважена сума. Користувач також має можливість установити власний метод дефазифікації. Для цього необхідно вибрати команду **Custom…** і в графічному вікні, що з'явилося, надрукувати ім'я функції, що реалізує цю операцію.

Редактор функцій приналежності (Membership Function Editor) призначений для завдання наступної інформації про термножини вхідних і вихідних змінних: кількість термів; найменування термів; тип і параметри функцій приналежності, що необхідні для представлення лінгвістичних змінних у вигляді нечітких множин.

Редактор функцій приналежності може бути викликаний з будь-якого GUI-модуля, використовуваного із системами нечіткого логічного виведення, командою Membership Functions... меню Edit або натисканням клавіш Ctrl+2.

У FIS-редакторі відкрити редактор функцій приналежності можна також подвійним кликом лівою кнопкою миші по полю вхідної або вихідної змінних. Загальний вигляд вікна редактора функцій приналежності з вказівкою функціонального призначення основних полів графічного вікна приведений на рис. 3.2.

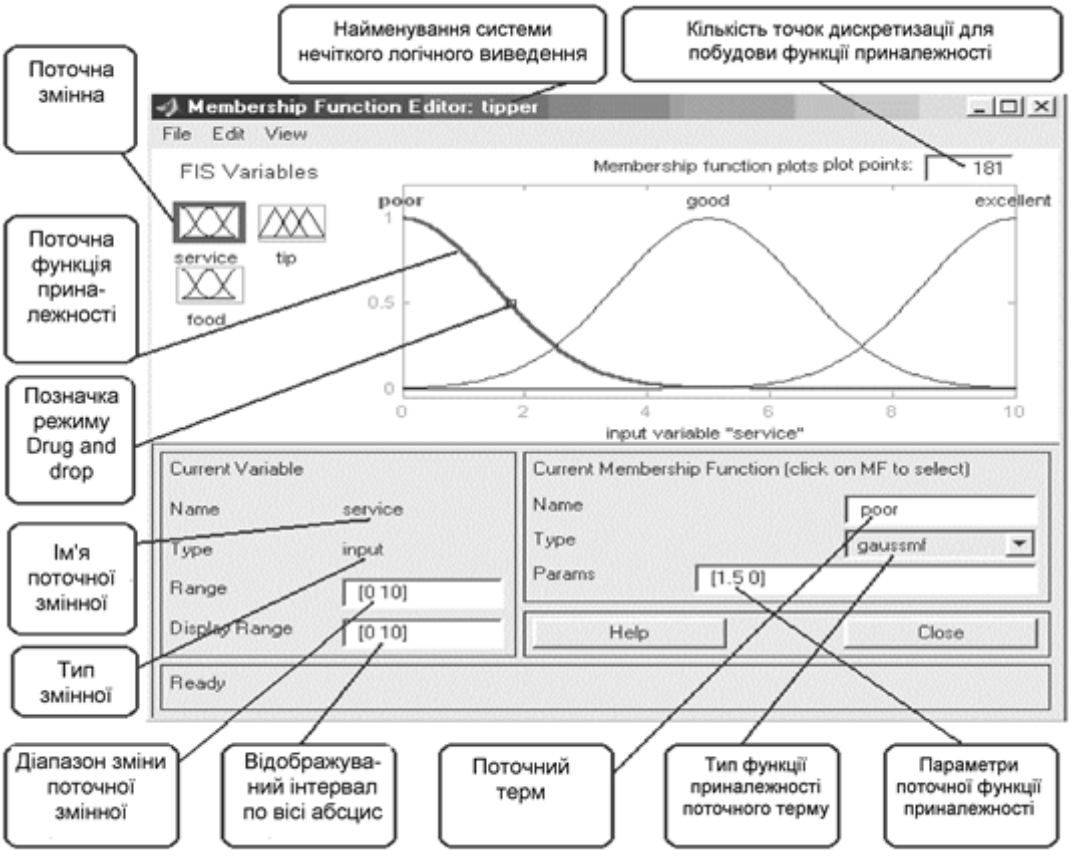

Рис. 3.2. Вікно редактора функцій приналежності

Редактор функцій приналежності містить чотири меню – **File**, **Edit**, **View**, **Type** і чотири вікна введення інформації – **Range**, **Display Range**, **Name** і **Params**. Ці чотири вікна призначені для завдання діапазону зміни поточної змінної, діапазону виведення функцій приналежності, найменування поточного лінгвістичного терму і параметрів його функції приналежності відповідно.

Параметри функції приналежності можна підбирати й у графічному режимі шляхом зміни форми функції приналежності за допомогою технології «Drug and drop». Для цього необхідно позиціонувати курсор миші на знаку режиму «Drug and drop», натиснути на ліву кнопку миші і, не відпускаючи її, змінювати форму функції приналежності. Параметри функції приналежності будуть перераховуватися автоматично.

Меню Edit: Команда Undo скасовує раніше зроблену дію. Виконується також по натисканню Ctrl+Z.

Команда **Add MFs…** дозволяє додати терми в терм-множину, використовувану для лінгвістичної оцінки поточної змінної. При виборі цієї команди з'явиться діалогове вікно, у якому необхідно вибрати тип функції приналежності і кількість термів. Значення параметрів функцій приналежності будуть встановлені автоматично таким чином, щоб рівномірно покрити область визначення змінної, заданої у вікні **Range**. При зміні області визначення у вікні Range параметри функцій приналежності будуть промасштабовані.

Команда **Add Custom MF…** дозволяє додати один лінгвістичний терм, функція приналежності якого відрізняється від вбудованих. Після вибору цієї команди з'явиться графічне вікно, у якому необхідно надрукувати лінгвістичний терм (поле **MF name**), ім'я функції приналежності (поле **M-File function name**) і параметри функції приналежності (поле **Parameter list**).

Команда **Remove Selected MF** видаляє поточний терм із терм-множини поточної змінної. Ознакою поточної змінної є червона окантовка її прямокутника. Ознакою поточного терму є червоний колір його функції приналежності. Для вибору поточного терму необхідно провести позиціонування курсору миші на графіку функції приналежності і зробити клік лівою кнопкою миші.

Команда **Remove All MFs** видаляє всі терми з терм-множини поточної змінної.

Команда **FIS Properties…** відкриває FIS-редактор. Ця команда може бути також виконана натисканням Ctrl+1.

Команда **Rules…** відкриває редактор бази знань. Ця команда може бути також виконана натисканням Ctrl+3.

Меню Type дозволяє установити тип функцій приналежності термів, використовуваних для лінгвістичної оцінки поточної змінної.

Редактор бази знань (Rule Editor) призначений для формування і модифікації нечітких правил. Редактор бази знань може бути викликаний з будь-якого GUI-модуля, використовуваного із системами нечіткого логічного виведення, командою **Rules…** меню **Edit** або натисканням клавіш Ctrl+3. У FIS-редакторі відкрити редактор бази знань можна також подвійним кликом лівою кнопкою миші по прямокутнику з назвою системи нечіткого логічного виведення, розташованого в центрі графічного вікна.

Загальний вид редактора бази знань із указівкою функціонального призначення основних полів графічного вікна приведений на рис. 3.3.

У нижній частині графічного вікна розташовані кнопки **Help** і **Close**, що дозволяють викликати вікно довідки і закрити редактор відповідно.

Редактор функцій приналежності містить чотири системних меню **File**, **Edit**, **View**, **Options**, меню вибору термів вхідних і вихідних змінних, поля установки логічних операцій ТА, АБО, НЕ і ваг правил, а також кнопки редагування і перегляду правил.

Для введення нового правила в базу знань необхідно за допомогою миші вибрати відповідну комбінацію лінгвістичних термів вхідних і вихідних змінних, установити тип логічного зв’язування (ТА або АБО) між змінними усередині правила, установити наявність чи відсутність логічної операції НЕ для кожної лінгвістичної змінної, увести значення вагового коефіцієнта правила і натиснути кнопку **Add Rule**. За замовчуванням установлені наступні параметри: логічне зв’язування змінних усередині правила – ТА; логічна операція НЕ – відсутня; значення вагового коефіцієнта правила – 1.

Можливі випадки, коли істинність правила не змінюється при довільному значенні деякої вхідної змінної, тобто ця змінна не впливає на результат нечіткого логічного виведення в даній області факторного простору. Тоді лінгвістичне значення цієї змінної необхідно установити як none.

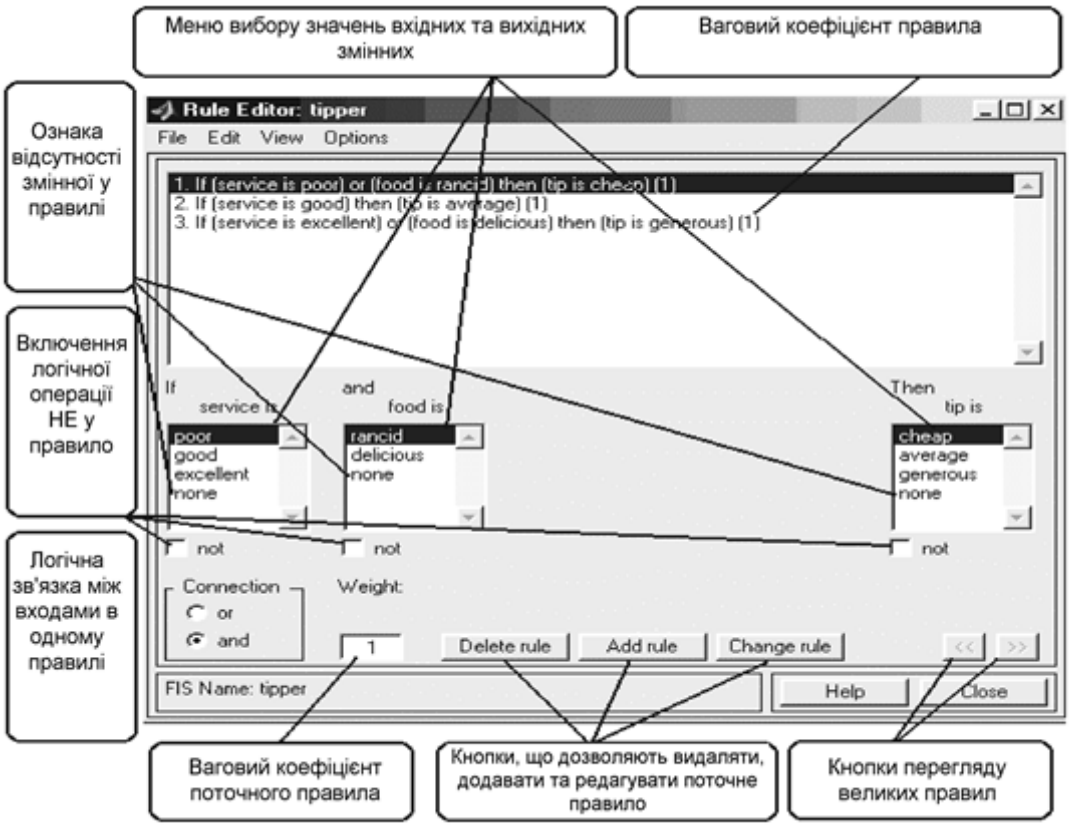

Рис. 3.3. Загальний вид редактора бази знань із вказівкою функціонального призначення основних полів графічного вікна

Для видалення правила з бази знань необхідно зробити однократний клик лівою кнопкою миші на цьому правилі та натиснути кнопку**Delete Rule**.

Для модифікації правила необхідно зробити однократний клик лівою кнопкою миші на цьому правилі, потім установити необхідні параметри правила і натиснути кнопку **Edit Rule**.

Меню редактора бази знань MatLab

Меню Edit: Команда **Undo** скасовує раніше зроблену дію. Виконується також по натисканню Ctrl+Z.

Команда **FIS Properties…** відкриває FIS-редактор. Ця команда може бути також виконана натисканням Ctrl+1.

Команда **Membership Function…** відкриває редактор функцій приналежностей. Ця команда може бути також виконана натисканням Ctrl+2.

Меню Options дозволяє установити мову і формат правил бази знань. При виборі команди Language з’явиться список мов

English (Англійська), Deutsch (Німецька), Francais (Французька), з якого необхідно вибрати одну.

При виборі команди **Format** з'явиться список можливих форматів правил бази знань: **Verbose** – лінгвістичний; **Symbolic** – логічний;**Indexed** – індексований.

Візуалізація нечіткого логічного виведення

Візуалізація нечіткого логічного виведення здійснюється за допомогою GUI-модуля **Rule Viewer**. Цей модуль дозволяє проілюструвати хід логічного виведення за кожним правилом, одержання результуючої нечіткої множини і виконання процедури дефазифікації. **Rule Viewer** може бути викликаний з будь-якого GUI-модуля, використовуваного із системами нечіткого логічного виведення, командою **View rules …** меню **View** чи натисканням клавіш Ctrl+4. Вигляд **Rule Viewer** для системи логічного виведення **tipper** із вказівкою функціонального призначення основних полів графічного вікна приведений на рис. 3.4.

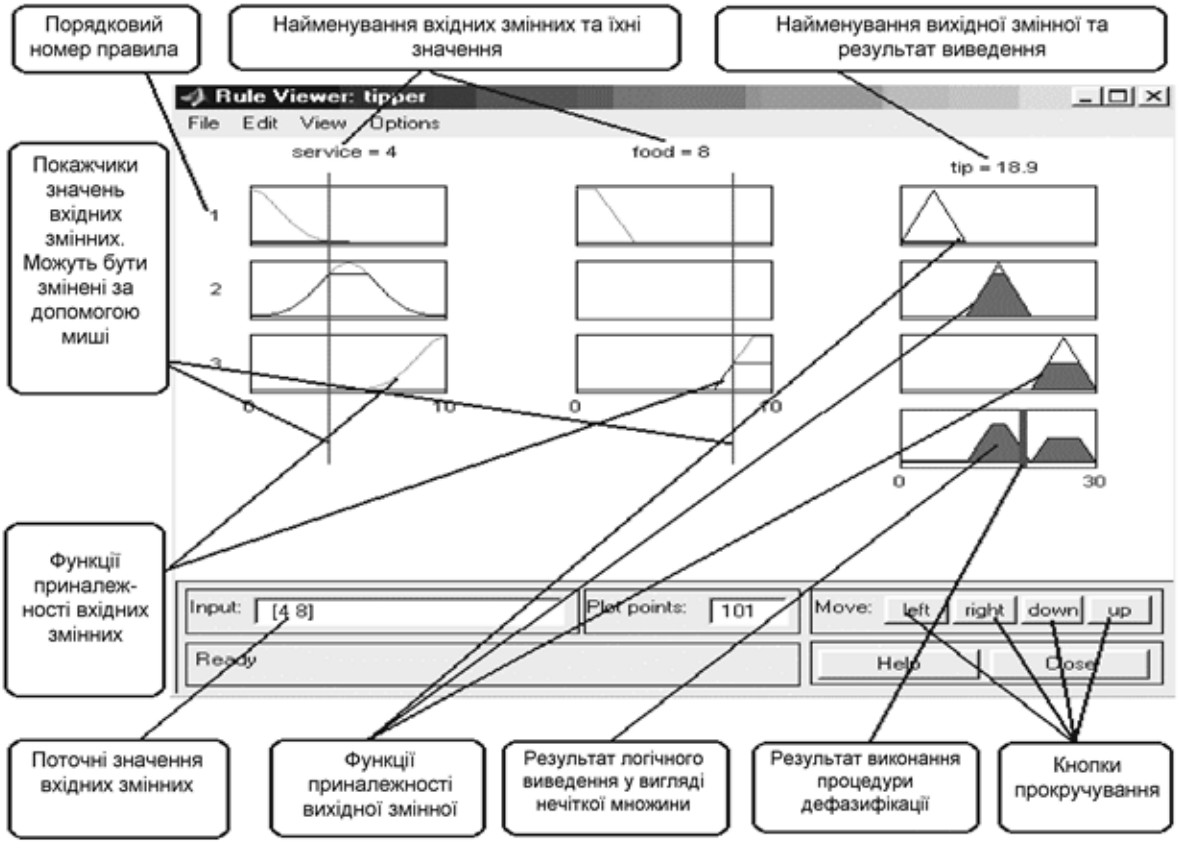

Рис. 3.4. Вікно модуля Rule Viewer

Rule Viewer містить чотири меню – **File**, **Edit**, **View**, **Options**, два поля введення інформації – **Input** і **Plot points** та кнопки прокручування зображення вліво – вправо (left-right), униз (up-down). У нижній частині графічного вікна розташовані також кнопки **Help** і**Close**, що дозволяють викликати вікно довідки і закрити редактор відповідно. Кожне правило бази знань представляється у виді послідовності горизонтально розташованих прямокутників. При цьому перші два прямокутники відображають функції приналежностей термів посилки правила (Якщо – частина правила), а останній третій прямокутник відповідає функції приналежності терму-наслідку вихідної змінної (То – частина правила).

Порожній прямокутник у візуалізації другого правила означає, що в цьому правилі посилка за змінною food відсутня (food is none). Жовте зафарбування графіків функцій приналежностей вхідних змінних вказує, наскільки значення входів відповідають термам даного правила. Для виведення правила у форматі **Rule Editor** необхідно зробити однократний клік лівої кнопки миші по номеру відповідного правила. У цьому випадку зазначене правило буде виведено в нижній частині графічного вікна.

Блакитне зафарбування графіка функції приналежності вихідної змінної являє собою результат логічного виведення у вигляді нечіткої множини за даним правилом. Результуючу нечітку множину, що відповідає логічному виведенню за всіма правилами, показано в нижньому прямокутнику останнього стовпця графічного вікна. У цьому ж прямокутнику червона вертикальна лінія відповідає чіткому значенню логічного виведення, отриманого в результаті дефазифікації.

Введення значень вхідних змінних може здійснюватися двома способами: шляхом уведення чисельних значень у поле **Input**; за допомогою миші, шляхом переміщення ліній-покажчиків червоного кольору.

В останньому випадку необхідно позиціонувати курсор миші на червоній вертикальній лінії, натиснути на ліву кнопку миші і, не відпускаючи її, перемістити покажчик на потрібну позицію. Нове чисельне значення відповідної вхідної змінної буде перелічено автоматично і виведене у вікно **Input**.

У поле **Plot points** задається кількість точок дискретизації для побудови графіків функцій приналежності. Значення за замовчуванням – 101.

Меню системи tipper

Меню Edit: Команда **FIS Properties…** відкриває FIS-редактор. Ця команда може бути також виконана натисканням Ctrl+1.

Команда **Membership Functions…** відкриває редактор функцій приналежності. Ця команда може бути також виконана натисканням Ctrl+2.

Команда **Rules…** відкриває редактор бази знань. Ця команда може бути також виконана натисканням Ctrl+3.

Меню Options містить тільки одну команду **Format**, що дозволяє установити один з наступних форматів виведення обраного правило в нижній частині графічного вікна: **Verbose** – лінгвістичний; **Symbolic** – логічний; **Indexed** – індексований.

Питання для самоконтролю

1. Що таке «нечітка логіка»?
2. Яке місце посідає нечітка логіка у сучасних інформаційних системах?
3. Що таке «нечітка множина»?
4. Що таке «лінгвістична змінна» та «терм-множина»?
5. Які операції існують у нечіткій логіці?
6. Що являє собою нечітке логічне виведення?
7. Які моделі нечіткого логічного виведення найбільш поширені?
8. Які різновиди функції приналежності реалізовані у MatLab?
9. Які моделі логічного виведення реалізовані у MatLab?
10. Який зміст мають процедури фазифікації та дефазифікації і як вони реалізуються у моделях логічного виведення Мамдані та Сугено?

ЛАБОРАТОРНА РОБОТА 3.1

Тема. Проектування систем нечіткого виводу на основі алгоритму Мамдані

Мета роботи: освоїти методику проектування системи нечіткого виводу на основі розробки та використання баз знань продукційних правил з використанням алгоритму Мамдані.

Теоретичні відомості

Знання можна формалізувати у вигляді системи нечітких логічних висловлювань. Кожне висловлювання можна оцінити нечітким ступенем істинності. Кожне таке висловлювання можна описати за допомогою відношень множин лінгвістичних нечітких змінних.

Лінгвістична змінна – це кортеж таких значень $\langle \beta, T, X, G, M \rangle$,

де β – ім'я змінної;

Т – базова множина значень її термів – значень, кожне з яких надається за допомогою нечіткої множини;

Х – множина – носій можливих конкретних значень змінної для всіх термів;

G – деяка синтетична процедура генерації нових термів з множини Т;

М – семантична процедура надання терму певної нечіткої змінної вигляду $\langle X, \mu_i(X) \rangle$, $\mu_i(X)$ – функція приналежності *i*-того терму з множини Т.

У системі **MATLAB** існує середовище для формування систем знань нечіткого виводу. Для входу в це середовище слід ввести в командному рядку слово **fuzzy** і натиснути клавішу **<Enter>**. Побудова *системи нечіткого виводу* (СНВ), яка основана на використанні алгоритму Мамдані, має такі етапи:

1. Проектування бази правил СНВ. Кожне правило представляється у вигляді:

Якщо <умова> **тоді** <заключення> [міра вірності правила]

Для алгоритму Мамдані <умова> і <заключення> виглядають як логічні зв'язки таких записів: **<нечітка змінна> = < значення >**

2. Введення цих правил у СНВ.

3. Використання СНВ для обробки вхідної інформації у вигляді конкретних значень вхідних (нечітких) змінних. Цей етап, в свою чергу, розкладається на такі складові:

3.1. Введення значень вхідних змінних. Тобто деяких фактів, які вважаються істинними на 100 %.

3.2. Фазифікація вхідних змінних – встановлення відповідності між конкретним значенням вхідних змінних і значенням її терму, разом з функцією приналежності.

3.3. Агрегування складних умов, які стоять у правилах після ключового слова ЯКЩО, тобто визначення степені істинності всіх умов в усіх правилах, якщо умови надаються за допомогою складних логічних виразів. Правило активується, якщо істинність його умови більша за нуль. У базах знань процедура агрегування умов у правилах виконується за допомогою нечітких логічних операцій – нечіткої кон'юнкції, нечіткої діз'юнкції, нечіткої відмови та ін.

4. Активація підзаключень – процес визначення степені істинності (приналежності до відповідних термів) змінних, які стоять у заключеннях активних правил, за формулою: $c_k = b_k F_k$, де c_k – степінь істинності заключення правила k, b_k – степінь істинності його умови, F_k – степінь істинності самого правила (ваговий коефіцієнт k-правила). Після визначення вектору $C = (c_1,..,c_q)$ визначаються функції приналежності для кожного із підзаключень для кожної вихідної лінгвістичної змінної. Припустимо, що відповідний терм вихідної лінгвістичної змінної визначається функцією приналежності $\mu(y)$. Тоді після процедури активації отримуємо поновлену функцію приналежності відповідного терму (підзаключення) $\mu'(y)$ за одним із методів нечіткої композиції:

- min – активізація: $\mu'(y) = \min\{c_i, \mu(y)\}$;

- prod-активізація: $\mu'(y) = c_i \mu(y)$;
- average-активізація: $\mu'(y) = 0.5(c_i + \mu(y))$.

Зазначимо, що різні правила підзаключень можуть містити однакові терми лінгвістичних змінних. У цьому випадку для кожного терму визначаємо множину різних функцій приналежності, які обчислюються за одним із правил нечіткої композиції по кожному правилу продукцій. Остаточна функція приналежності для цього терму визначається у наступному пункті.

5. Акумуляція заключень, тобто визначення значення функцій приналежності для термів усіх вихідних змінних. Якщо для одного терму визначена множина функцій приналежності $\mu_1'(y),...,\mu_p'$, то акумуляція виконується за одним із правил об'єднання нечітких множин:

- об'єднання: $\mu'(y) = \max\{\mu_1'(y), \mu_2'\}$;
- алгебраїчне об'єднання: $\mu'(y) = \mu_1'(y) + \mu_2'(y) - \mu_1'(y)\mu_2'(y)$;
- граничне об'єднання: $\mu'(y) = \max\{\mu_1'(y) + \mu_2'(y) - 1, 0\}$;
- операція λ-суми: $\mu'(y) = \lambda\mu_1'(y) + (1-\lambda)\mu_2'(y)$, $\lambda \in [0,1]$.
- драстичне об'єднання: $\mu'(y) = \begin{cases} \mu_1'(y), & if\ \mu_2'(y) = 0, \\ \mu_2'(y), & if\ \mu_1'(y) = 0, \\ 1, & else. \end{cases}$

6. Дефазифікація вихідних змінних (визначення конкретних значень за функціями приналежності термів) розглядається методом центру ваги для неперервних та дискретних нечітких множин за формулами:

$$z = \frac{\int_{y_{\min}}^{y_{\max}} y\mu'(y)dy}{\int_{y_{\min}}^{y_{\max}} \mu'(y)dy}, \quad z = \frac{\sum_{i=1}^{n} y_i \mu'(y_i)}{\sum_{i=1}^{n} \mu'(y_i)}.$$

Розглянемо принципи побудови та роботи системи нечіткого виводу на прикладі задачі візуалізації поверхні, яка задана функцією.

Завдання 1. Проектування та використання системи нечіткого виводу

За допомогою СНВ зобразити поверхню функції $y = (x_1^2 - 8)\cos(x_2)$ на множині $x_1 \in [0,4]; x_2 \in [0,4]$.

Методичні рекомендації

Проектування системи нечіткого виводу слід проводити на основі графічного зображення вказаної залежності. Для цього в М-файлі складемо таку програму:

```
%Побудова графіка функції y=(x1^2-8)*cos(x2)
%в області x1є[0,4] и x2є[0,4].
n=15;
x1=0:4/(n-1):4;
x2=0:4/(n-1):4;
y=zeros(n,n);
for j=1:n
y(j,:)=(x1.^2-8)*cos(x2(j));
end
surf(x1,x2,y)
xlabel('x1')
ylabel('x2')
zlabel('y')
title('Target').
```

У результаті виконання цієї програми отримуємо графічне зображення, яке наведено на рис. 3.1.1.

Проектування СНВ складається з таких кроків.

Крок 1. Завантажити основний fis-редактор у (редактор нечіткого виводу) введенням у командному рядку слова **fuzzy.** Після чого з'явиться вікно редактора нечіткого виводу.

Крок 2. Ввести нову вхідну змінну. Для цього вибрати пункт **Add Input** в меню **Edit**.

Крок 3. Перейменувати першу вхідну змінну. Для цього слід зробити одне натиснення лівої кнопки миші на блоці **Input1**,

ввести нове позначення **x1** у поле редагування імені поточної змінної і натиснути **<Enter>**.

Крок 4. Перейменувати другу вхідну змінну. Для цього зробити одне натиснення лівої кнопки миші на блоці **input2**, ввести нове позначення **x2** у поле редагування імені поточної змінної і натиснути **<Enter>**.

Крок 5. Перейменувати вихідну змінну. Для цього зробити одне натиснення лівої кнопки миші на блоці **output1**, ввести нове позначення **y** в поле редагування імені поточної змінної і натиснути **<Enter>**.

Крок 6. Задати ім'я системі. Для цього в меню **File** вибрати в підменю **Export to disk** і ввести ім'я файлу, наприклад, **first**.

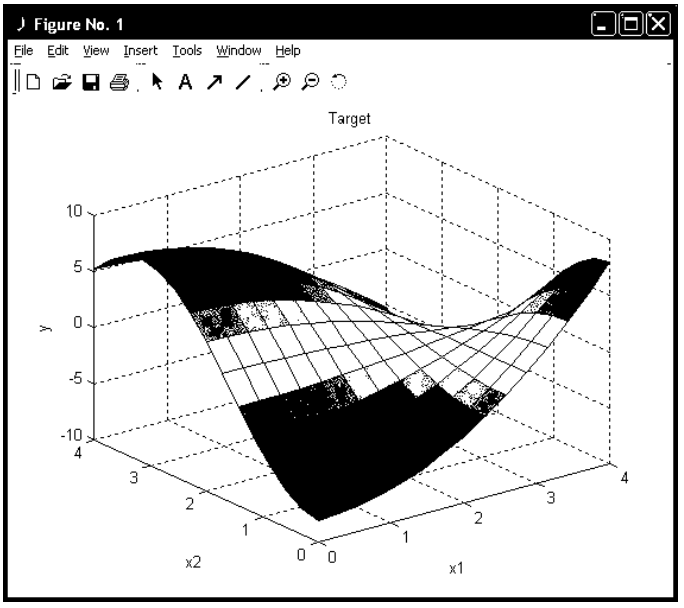

Рис. 3.1.1. Зображення поверхні функції

Крок 7. Перейти в редактор функцій приналежності. Для цього зробити швидко подвійне натиснення лівої кнопки миші на блоці **x1**.

Крок 8. Задати діапазон зміни змінної **x1**. Для цього надрукувати **0 4** в поле **Range** і натиснути **<Enter>**.

Крок 9. Задати функції приналежності змінної **x1**. Для лінгвістичної оцінки цієї змінної будемо використовувати 3 терми з трикутними функціями належності. Якщо в вікні немає ще функцій належності, тоді в меню **Edit** слід вибрати команду **Add MFs...** У результаті з'явиться діалогове вікно вибору типу і кількості функцій

приналежності. За замовченням ці 3 терми мають трикутну функцію приналежності. Тому просто потрібно натиснути **<Enter>.**

Крок 10. Задати *найменування термів змінної* **x1**. Для цього робимо одне натиснення лівою кнопкою миші на графіку першої функції приналежності (див. рис. 3.2). Потім вводимо найменування терму, наприклад, **L** (**Низький)**, у полі **Name** і натискаємо **<Enter>**. Потім робимо одне натиснення лівою кнопкою миші на графіку другої функції приналежності і вводимо найменування терму, наприклад, **A** (**Середній)**, в полі **Name і** натискаємо **<Enter>**. Ще раз робимо одне натиснення лівою кнопкою миші по графіку третьої функції приналежності і введемо найменування терму, наприклад, **H** (**Високий)**, в полі **Name** і натискаємо **<Enter>**. У результаті отримуємо графічне вікно, яке зображено на рис. 3.1.2.

Крок 11. Задамо функції приналежності змінної **x2**. Для лінгвістичної оцінки цієї змінної будемо використовувати 5 термів з гаусовськими функціями приналежності. Для цього активізуємо змінну **x2** за допомогою натиснення лівої кнопки миші на блоці **x2.** Задамо діапазон змін **x2**. Для цього надрукуємо 0 **4** в полі **Range** (рис. 3.3) і натиснемо **<Enter>**. Потім у меню **Edit** виберемо команду **Add MFs...**. У діалоговому вікні, що з'явиться, оберемо тип функції приналежності **gaussmf** в полі **MF type** і **5** термів у полі **Number of MFs**. Після чого натискаємо <**Enter>.**

Крок 12. За аналогією з кроком *10* задамо такі найменування термів змінної **x2**: **L (Низький), LA (Нижче середнього), A (Середній), HA (Вище середнього), H (Високий)**. У результаті отримуємо графічне вікно, яке зображене на рис. 3.1.3.

Крок 13. Задамо функції приналежності змінної **y**. Для лінгвістичної оцінки цієї змінної будемо використовувати 5 термів з трикутними функціями приналежності. Для цього активуємо змінну **y** за допомогою натиснення лівої кнопки миші на блоці **y.** Задамо діапазон змін змінної **y**. Для цього надрукуємо – **10 10** в полі **Range** (рис. 3.4) і натиснемо **<Enter>**. Потім у меню **Edit** оберемо команду **Add MFs...**. У діалоговому вікні, що з'явиться, виберемо **5** термів у полі **Number of MFs**. Після чого натискаємо **<Enter>.**

Крок 14. За аналогією з кроком *10* задамо такі найменування термів змінної **y**: **L** (**Низький)**, **LA** (**Нижче середнього) A (середній)**, **HA (Вище середнього)**, **H (Високий)**. У результаті отримуємо графічне вікно, яке представлене на рис. 3.1.4.

Крок 15. Перейдемо в редактор бази знань **RuleEditor**. Для цього оберемо в меню **Edit** команду **Rules** або в меню **View** команду **Edit rules...**.

Крок 16. На основі візуального спостереження за графіком, який зображений на рис. 1.1, сформуємо такі правила:

1. **Якщо** x1 = Низький **і** x2 = Низький, **тоді** y = Низький.
2. **Якщо** x1 = Низький **і** x2 = Високий, **тоді** y = Середній.
3. **Якщо** x1 = Низький **і** x2 = Вище середнього, **тоді** y = Високий.
4. **Якщо** x1 = Високий **і** x2 = Низький, **тоді** y = Вище середнього.
5. **Якщо** x1 = Високий **і** x2 = Високий, **тоді** y = Низький.
6. **Якщо** x1 = Середній **і** x2 = Середній, **тоді** y = Середній.
7. **Якщо** x1 = Середній **і** x2 = Вище середнього, **тоді** y = Вище середнього.
8. **Якщо** x1 = Низький **і** x2 = Нижче середнього, **тоді** y = Нижче середнього.
9. **Якщо** x1 = Середній **і** x2 = Вище середнього, **тоді** y = Середній.

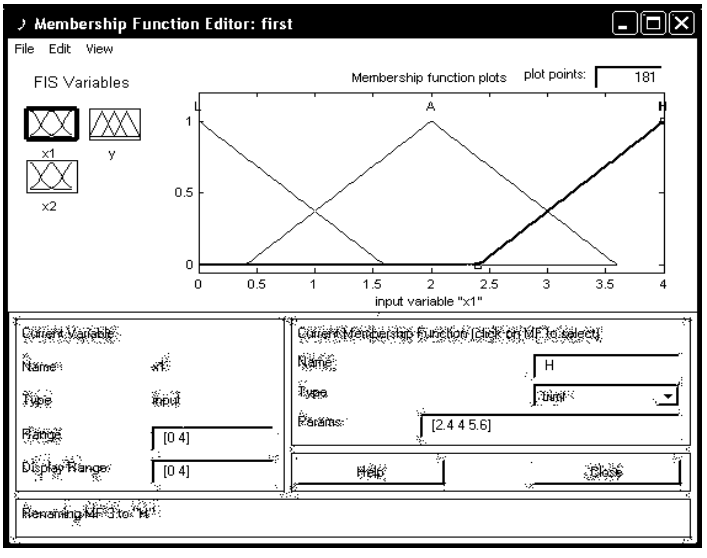

Рис. 3.1.2. Функція приналежності змінної x_1

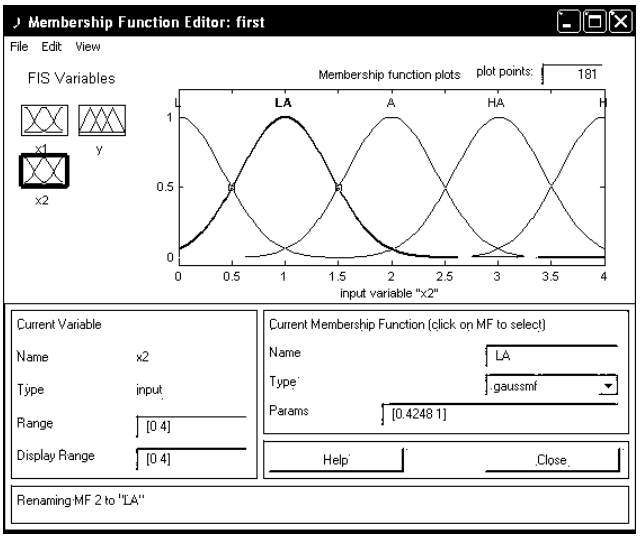

Рис. 3.1.3. Функція приналежності змінної x_2

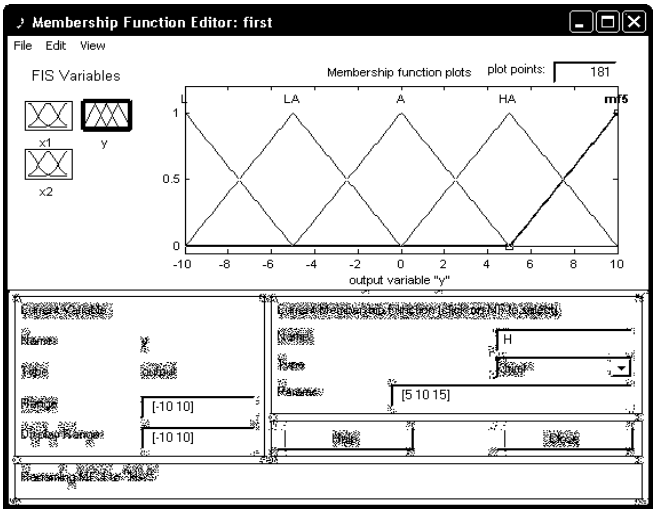

Рис. 3.1.4. Функції приналежності змінної y

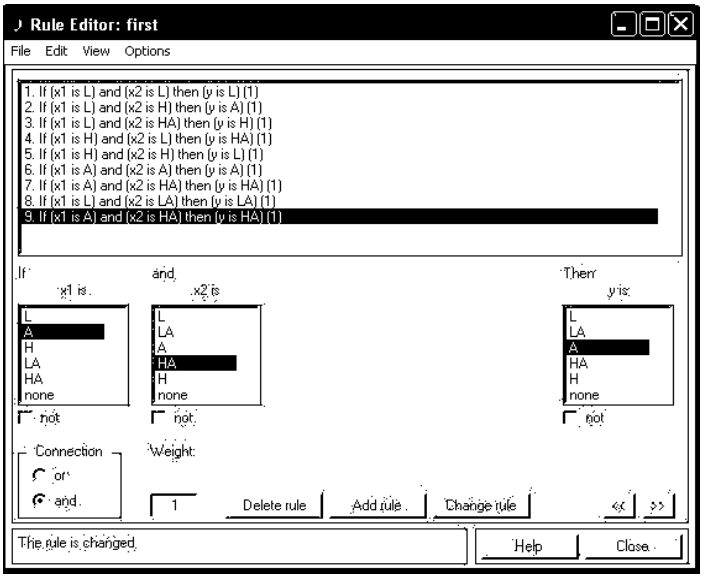

Рис. 3.1.5. Вікно правил

Для введення правила необхідно обрати в меню відповідну комбінацію термів і натиснути кнопку **Add rule**. На рис. 3.1.5 зображено вікно редактора бази знань після введення усіх 9 правил. Число в дужках в кінці кожного правила являє собою вагові коефіцієнти відповідного правила. Потрібно звернути увагу на параметр **Weight**, який вказує вагу нечіткої впевненості у правилі. Його можна задавати в діапазоні $[0,1]$.

Крок 17. Збережемо побудовану систему. Для цього в меню **File** в підменю **Export** оберемо команду **To disk**.

На рис. 3.1.6 приведено вікно візуалізацій нечіткого логічного виводу. Це вікно активується командою **View rules...** меню **View.** У полі **Input** вказуються значення вхідних змінних, для яких виконується логічний вивід. Тобто обраховується за алгоритмом Мамдані значення вихідної змінної.

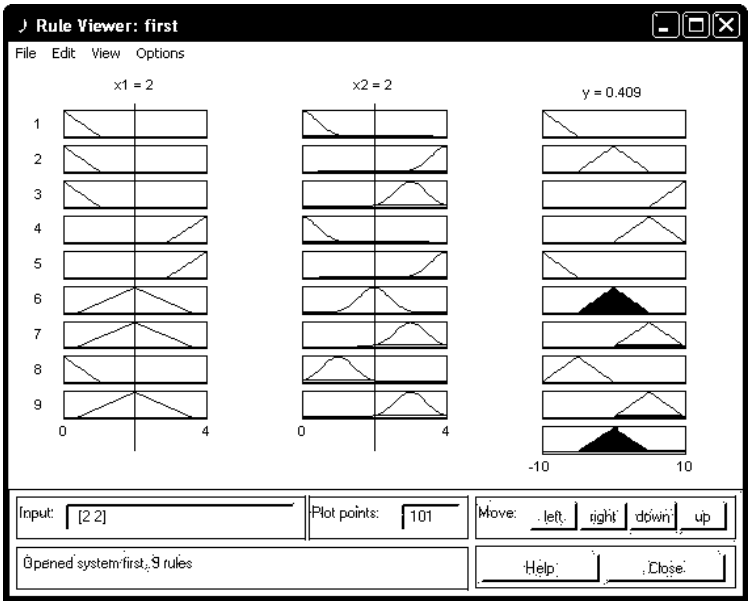

Рис. 3.1.6. Візуалізація нечіткого виводу

На рис. 3.1.7 приведена поверхня «входи-вихід», яка відповідає синтезованій системі логічного виводу. Для виводу цього вікна необхідно використати команду **View surface...** меню **View.** Порівнюючи поверхні на рис. 3.1.1 і рис. 3.1.7, можна зробити висновок, що нечіткі правила досить добре описують складну нелінійну залежність.

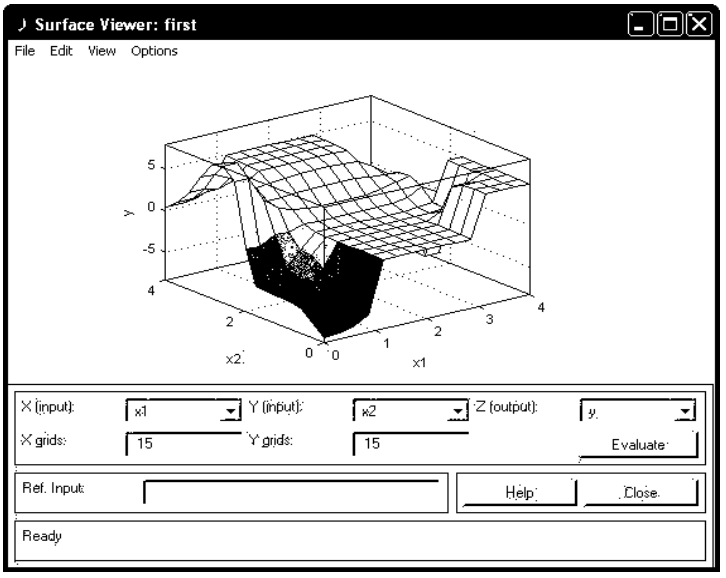

Рис. 3.1.7. Поверхня системи «вхід-вихід»

Завдання для самостійного виконання

1. Створити систему нечіткого виводу, яка моделює залежність $y = x_1^2 \sin(x_2 - 1)$ при $x_1 \in [-7,3]; x_2 \in [-4.5,2]$.

2. Створити систему нечіткого виводу, яка відтворює поверхню $y = \ln(x_1 + 1)\cos(x_2)$ при $x_1 \in [1,5]; x_2 \in [0,2]$.

3. Створити систему нечіткого виводу, яка відтворює поверхню $y = \ln(x_1 + 1)\dfrac{1}{1 + x_2}$ при $x_1 \in [1,5]; x_2 \in [0,2]$.

ЛАБОРАТОРНА РОБОТА 3.2

Тема. Проектування систем нечіткого виводу на основі алгоритму Сугено

Мета роботи: засвоїти методику проектування системи нечіткого виводу на основі розробки та використання баз знань продукційних правил з використанням алгоритму Сугено. Провести порівняльний аналіз алгоритмів Мамдані і Сугено.

Теоретичні відомості

Розглянемо основні етапи проектування систем нечіткого виводу за алгоритмом Сугено на прикладі задачі візуалізації

поверхні (рис. 3.2.1), яка реалізує залежність $y = (x_1^2 - 8)\cos x_2$ на відрізку $x_1 \in [0,4]; x_2 \in [0,4]$. Відмінність СНВ алгоритму Сугено полягає у проектуванні вихідних змінних. Формування бази правил систем нечіткого виводу такого формату:

Правило <#>: **Якщо** «змінна_1 = значення_А» **і** «змінна_2 = значення_В» **Тоді** «$y = k_1 A + k_2 B + k_0$»

або

Правило <#>: **Якщо** «змінна_1 = значення_А» **і** «змінна_2 = значення_В» **Тоді** «змінна_y = значення_С».

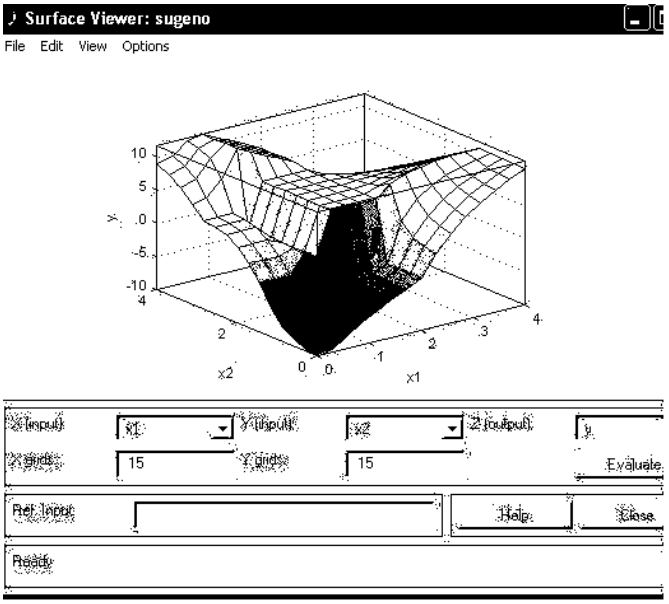

Рис. 3.2.1. Вихідна поверхня

Моделювання заданої поверхні будемо реалізовувати за допомогою таких правил бази знань:

1. Якщо x1 та x2 низькі, тоді $y = -10$.
2. Якщо x1 низьке, тоді $y = 3{,}75x_2 - 10$.
3. Якщо x1 низьке і x2 вище середнього, тоді $y = 7$.
4. Якщо x1 та x2 високі, тоді $y = -10$.
5. Якщо x2 низьке, тоді $y = 4x_1 - 10$.
6. Якщо x1 високе, тоді $y = 15 - 3{,}75x_2$.
7. Якщо x2 високе, тоді $y = 15 - 3{,}75x_1$.
8. Якщо x1 низьке і x2 високе, тоді $y = 7$.

9. Якщо x1 середнє і x2 середнє, тоді $y = 7$.

10. Тоді проектування системи нечіткого виводу типу Сугено лежить у виконанні такої послідовності кроків.

Крок 1. Для завантаження основного fis-редактору надрукуємо слово **fuzzy** у командному рядку Mathlab.

Крок 2. Оберемо тип системи. Для цього в меню **File** в підменю **New fis…** оберемо команду **Sugeno**.

Крок 3. Додамо другу вхідну змінну. Для цього в меню **Edit** оберемо команду **Add input**.

Крок 4. Перейменуємо першу вхідну змінну. Для цього зробимо одне натиснення лівої кнопки миші на блоці **input1**, введемо нове позначення **x1** в полі редагування імені поточної змінної і натиснемо <**Enter**>.

Крок 5. Перейменуємо другу вхідну змінну. Для цього зробимо одне натиснення лівою кнопкою миші на блоці **input2**, введемо нове позначення **x2** в полі редагування імені поточної змінної і натиснемо <**Enter**>.

Крок 6. Перейменуємо вихідну змінну. Для цього зробимо одне натиснення лівою кнопкою миші на блоці **output1**, введемо нове позначення **y** в полі редагування імені поточної змінної і натиснемо <**Enter**>.

Крок 7. Задамо ім’я системи. Для цього в меню **File** в підменю **Export** оберемо команду **To disk** і введемо ім’я файлу, наприклад, **FirstSugeno**.

Крок 8. Перейдемо в редактор функцій приналежності. Для цього зробимо швидке подвійне натиснення лівої кнопки миші на блоці **x1**.

Крок 9. Задамо діапазон змін змінної **x1**. Для цього надрукуємо **0 4** в полі **Range** і натиснемо <**Enter**> (рис. 3.2.2).

Крок 10. Задамо функції приналежності змінної x_1. Для лінгвістичної оцінки цієї змінної будемо використовувати 3 терми з трикутними функціями приналежності, які встановлені за замовченням. Задамо найменування термів змінної x_1. Для цього робимо одне натиснення лівою кнопкою миші по графіку першої функції приналежності.

Крок 11. *Аналогічно з*адамо функції приналежності змінної x_2. Для лінгвістичної оцінки цієї змінної будемо використовувати 4 терми з трикутними функціями приналежності. Для цього активізуємо змінну x_2 за допомогою натиснення лівої кнопки миші на блоці x_2**.** Задамо діапазон змін x_2. Для цього надрукуємо **0 4** в полі **Range** і натиснемо **<Enter>**. Задамо найменування 4 термів (наприклад, **L Низький**, **A Середній**, **HA Вище середнього**, **H Високий**).

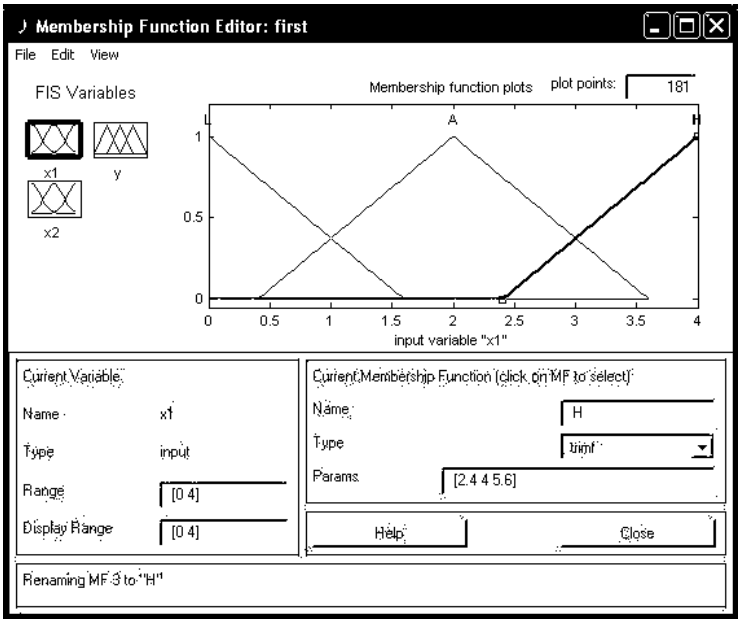

Рис. 3.2.2. Функції приналежності змінної x_1

Крок 12. Задамо лінійні залежності між входами і виходом, які наведено в базі знань (рис. 3.2.3). Для цього активуємо змінну **y** за допомогою натиснення лівої кнопки на блоці **y.** У правому верхньому куті можуть з’явитися позначення функцій приналежності, кожна з яких відповідає одній лінійній залежності між входами і виходом. У базі знань, яка наведена на початку файлу, вказані 6 різних залежностей: y = -10; y = 7; y = 3,75 x_1 -10; y = 4 x_1 -10;

y = -3,75 x_1 +15; y = -3,75 x_2 +15. Додамо ще необхідну кількість функцій залежності шляхом обирання команди **Add Mfs…** меню **Edit.**

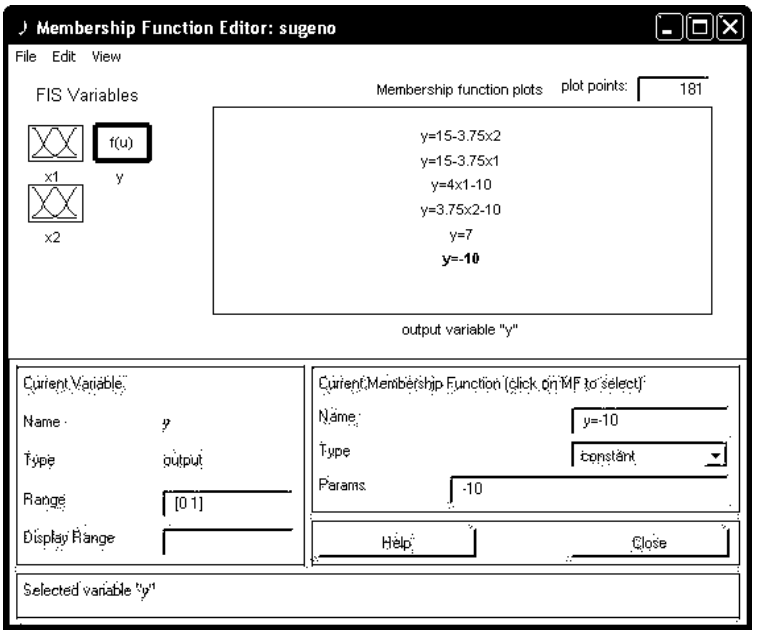

Рис. 3.2.3. Вікно лінійних залежностей «входи-вихід»

Крок 13. Задамо найменування і параметри цих залежностей. Для цього робимо одне натиснення лівою кнопкою миші по імені першої залежності **mf1**. Потім друкуємо назву залежності, наприклад **y = -10**, в полі **Name**, і встановлюємо тип залежності – константа шляхом обирання опції **Constant** в меню **Type**. Після цього вводимо значення параметру – **-10** в полі **Params**. Аналогічну процедуру робимо для другої змінної **y = 7.**

Для третьої функції **mf3** введемо найменування, наприклад, **y = 3.75 x1-10**. Потім вкажемо лінійний тип залежності шляхом вибору опції **Linear** в меню **Type** і введемо параметри залежності **3.75 0 -10** в полі **Params**. Для лінійної залежності порядок параметрів такий: перший параметр – коєфіцієнт при першій змінній, другий – при другій та ін., останній параметр – вільний член залежності. Таким же чином введемо назви і параметри для всіх 6 функцій приналежності змінної **y**.

У результаті отримуємо графічне вікно, яке представлено на рис. 3.2.4.

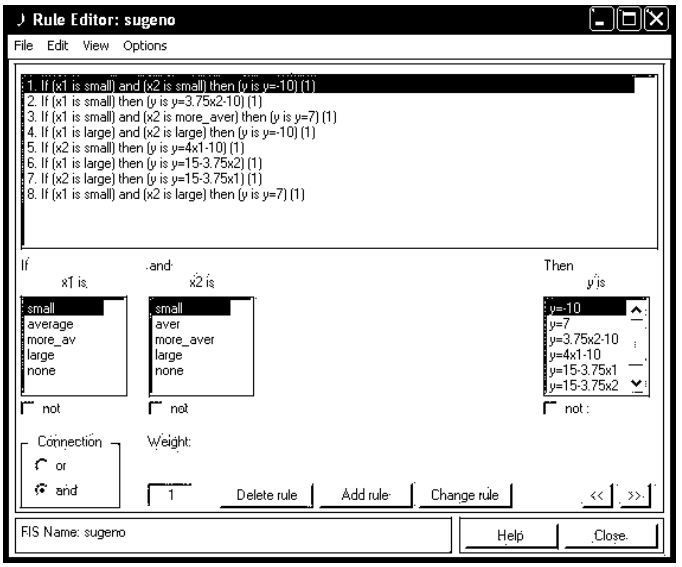

Рис. 3.2.4. Нечітка база знань для системи типу Сугено

Крок 14. Перейдемо в редактор бази знань **RuleEditor**. Для цього оберемо в меню **Edit** команду **Edit rules...** і введемо правила. Для вводу правила необхідно обрати відповідну комбінацію термів і залежностей і натиснути кнопку **Add rule**. На рис. 3.2.3 зображене вікно редактора бази знань після введення усіх 6 правил.

На рис. 3.2.5 приведено вікно візуалізації нечіткого логічного виводу. Це вікно активується командою **View rules...** меню **View.** У полі **Input** вказуються значення вхідних змінних, для яких виконується логічний вивід. Як можна побачити з рисунка, значення вихідної змінної розраховується як середнє зважене значення результатів виходу за кожним правилом.

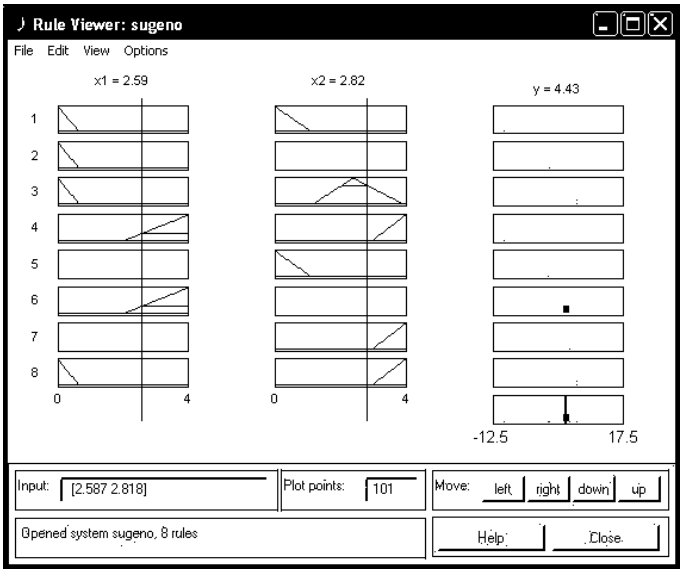

Рис. 3.2.5. Візуалізація нечіткого логічного виводу для системи типу Сугено

На рис. 3.2.6 приведена поверхня «входи-вихід», яка відповідає синтезованій нечіткій системі. Для виведення цього вікна необхідно використати команду **View surface...** меню **View.** Порівнюючи цю поверхню і поверхню на рис. 3.2.1, можна зробити висновок, що нечіткі правила достатньо добре описують складну нелінійну залежність. При цьому модель типу Сугено більш точна. Перевага моделей типу Мамдані полягає у тому, що правила бази знань є прозорі й інтуїтивно зрозумілі, тоді як для моделей типу Сугено не завжди зрозуміло, які лінійні залежності «входи-вихід» необхідно використовувати.

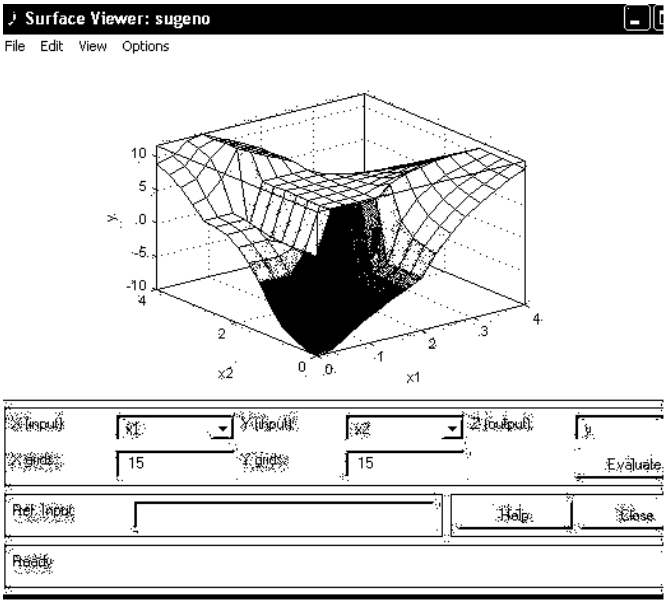

Рис. 3.2.6. Поверхня «вхід-вихід» для системи алгоритму Сугено

Завдання для самостійного виконання

1. Створити систему нечіткого виводу Сугено, яка моделює залежність $y = x_1^2 \sin(x_2 - 1)$ при $x_1 \in [-7,3]$, $x_2 \in [-4.5,2]$.

2. Створити систему нечіткого виводу Сугено, яка відтворює поверхню $y = \ln(x_1 + 1)\cos(x_2)$ при $x_1 \in [1,5]$, $x_2 \in [0,2]$.

3. Створити систему нечіткого виводу Сугено, яка відтворює поверхню $y = \ln(x_1 + 1)\dfrac{1}{1 + x_2}$ при $x_1 \in [1,5]$, $x_2 \in [0,2]$.

ЛАБОРАТОРНА РОБОТА 3.3

Тема. Розробка нечітких моделей систем підтримки прийняття рішень у соціально-економічних системах

Мета роботи: освоїти методику проектування і побудови систем підтримки прийняття рішень на основі моделі нечіткого виводу та бази знань нечітких продукцій на прикладі оцінювання ефективності фінансування інноваційного проекту. Основою прийняття рішення є бізнес-план проекту, у якому зазначені соціальне значення нововведення, ступінь ризику та оцінка чистого приведеного доходу як міра фінансової ефективності. На прикладі ознайомимося з особливостями розробки програмних засобів нечіткого моделювання.

Теоретичні відомості

1. Змістовна постановка задачі оцінювання ефективності інновації

Під інновацією (нововведенням) розуміють кінцевий результат інноваційної діяльності, що одержав реалізацію у вигляді нового або удосконаленого продукту, реалізованого на ринку, нового або удосконаленого технологічного процесу, використовуваного у практичній діяльності.

Інновація являє собою результат, отриманий від вкладення капіталу в нову техніку або технологію, у нові форми організації виробництва праці, обслуговування, управління тощо.

Процес створення, освоєння і поширення інновацій називається **інноваційною діяльністю** або інноваційним процесом.

Залежно від технологічних параметрів інновації підрозділяються на продуктові і процесові (технологічні). Продуктові інновації включають застосування нових матеріалів, нових напівфабрикатів і комплектуючих; одержання принципово нових продуктів. Процесні інновації означають нові методи організації виробництва (нові технології).

Інноваційний процес являє собою послідовність дій із розробки нових продуктів і операцій, з їхньої реалізації на ринку і з подальшого поширення результатів.

Інноваційний процес складається з таких основних кроків:

- зародження ідеї інновації;
- обґрунтування необхідності інноваційної зміни;
- оцінка ефективності інновації;
- розробка та технічна реалізація інноваційної ідеї;
- реалізація інновації в поведінці системи;
- просування інновації на ринку вимог.

Модель інноваційної діяльності в економіці можна представити у вигляді системи управління зі зворотнім зв’язком, як показано на рис. 3.3.1.

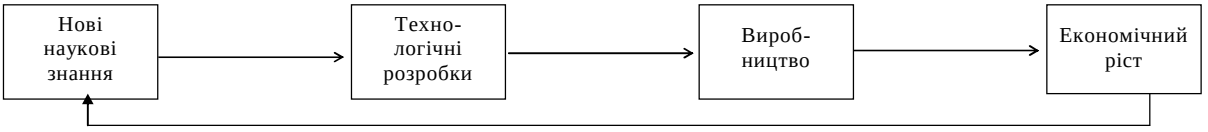

Рис. 3.3.1. Функціональна послідовність інноваційного процесу в економічній системі

Згідно з цією концепцією нові знання, одержані в результаті фундаментальних, а потім і прикладних досліджень, народжують автоматично нові технології і продукти. Тому нові знання повинні автоматично вести до економічного зростання.

Кожна стадія потребує фінансових та інших витрат на своє здійснення. Суб’єкта, який витрачає свої кошти і зусилля на проведення інноваційної діяльності, будемо називати інвестором. Звичайно, інвестор зацікавлений у тому, щоб інноваційна діяльність була ефективною.

Ефективність інноваційної зміни виражається через різноманітні показники. В умовах ринкових відносин не може бути уніфікованої системи показників. Кожен інвестор самостійно визначає цю систему, виходячи з особливостей інноваційного проекту, професіоналізму фахівців і менеджерів, й інших факторів.

Впровадження нововведень може дати такі види ефекту: науковий (приріст інформації); продуктовий (покращення якості товару); технологічний (зростання продуктивності праці, покращення умов праці); функціональний (зростання ефективності управління організацією); соціальний (покращення умов життя); екологічний (покращення умов довкілля).

Неекономічні ефекти теж несуть у собі потенційний економічний ефект. Наприклад, розроблений винахід як нововведення вищого рівня може дати економічний ефект або після його продажу, або після реалізації товару, розробленого на основі винаходу. Зниження викидів шкідливих компонентів в атмосферу, ґрунт, воду зберігає екосистему, збільшує тривалість життя людини тощо. Цей ефект неможливо відразу перевести в прибуток. Приведені приклади дозволяють зробити такий висновок: економічний ефект розробки, впровадження (перетворення в інновацію) або продажу нововведень може бути потенційним або фактичним (реальним, комерційним), а науково-технічний, соціальний і екологічний ефекти можуть мати форму тільки потенційного економічного ефекту. По суті, якщо взяти до уваги тільки кінцеві результати впровадження або продажу нововведень, будь-який вид інноваційної діяльності можна оцінити у вартісному вираженні. Критеріями кінцевої оцінки тут є: час одержання фактичного економічного ефекту і ступінь невизначеності його одержання (або **рівень ризику** вкладення інвестицій в інновації).

Не всі оцінки ефективності інновацій можна оцінити чітко. Наприклад, як можна визначити, що ступінь ризику вкладання коштів у деякий проект є високою і варто ще раз все зважити перед прийняттям рішення про його фінансування? Саме в такому випадку можна скористатися теорією нечітких продукцій.

З цієї причини може бути запропонована ідея розробки нечіткої моделі для оцінювання ефективності фінансування інноваційного проекту. При цьому як нечітка модель використовується система нечіткого виводу з вхідними і вихідною змінними, які розглянуті далі.

2. Опис вхідних і вихідних змінних задачі

Змістовна інтерпретація нечіткої моделі припускає вибір і специфікацію вхідних і вихідних змінних відповідної системи нечіткого виводу. При цьому в нечіткій моделі пропонується використовувати три вхідні змінні й одну вихідну.

Як *першу вхідну змінну* розглянемо ступінь ризику інноваційного проекту, що фінансується. Звичайно, чим вище ступінь ризику, тим більше втрачає сенс інвестувати даний проект.

Друга вхідна змінна визначає соціальне значення нововведення, його вплив на життя суспільства. Іноді високе соціальне значення може «компенсувати» помірний і середній ризик чи невеликий економічний ефект проекту.

Як *третю вхідну змінну* визначимо таку кількісну оцінку економічної ефективності інноваційного проекту, як чистий приведений дохід (ЧПД, NPV), який за умови дискретного характеру параметру часу життєвого циклу може бути представлений формулою:

$$NPV = \sum_{t=0}^{T} \frac{P(t) - IV(t)}{(1+q)^t},$$

де $P(t)$ – функція чистого прибутку від діяльності по розробці та реалізації нововведення протягом часу життєвого циклу проекту $t \in [0, T]$, $IV(t)$ – функція витрат проекту, які пов'язані з погашенням інвестиційних боргів, q – ставка дисконту (знецінення) грошових одиниць протягом кроку часу.

Вважається, що чим більше додатне значення має величина ЧПД, тим ефективнішим є проект, який фінансується.

Вихідною змінною виступає оцінка проекту для інвестора з точки зору ефективності його фінансування, що є основою прийняття рішень про виділення кредитів для реалізації нововведення.

Базуючись на досвіді й отриманих знаннях про ринок інноваційних проектів, інвестор застосовує такі евристичні правила:

1. Якщо ризик є допустимим, соціальне значення і ЧПД проекту є доволі високими, тоді приймається рішення про фінансування даного проекту. Тобто можна вважати, що в такому «ідеальному» випадку ступінь фінансування проекту є 100 %.

2. Якщо ризик є допустимим, соціальне значення середнє, тоді міра прийняття рішення про фінансування проекту збігається з нормованою величиною ЧПД, тобто відношенням ЧПД до свого максимально можливого значення: $\frac{NPV}{NPV_{\max}}$.

3. Якщо ризик є малим, соціальне значення посереднє, тоді міра прийняття рішення про фінансування проекту менша ніж нормована міра ЧПД раза в три.

4. Якщо ризик є малим, а соціальне значення є високим, тоді міра прийняття рішення про фінансування проекту збігається з нормованою величиною ЧПД.

5. Якщо ризик є малим, а соціальне значення є нижче за середній рівень, тоді міра прийняття рішення про фінансування проекту менше ніж нормована величина ЧПД у разів п’ять.

6. Якщо ризик є середнім, соціальне значення і ЧПД великі, тоді міра прийняття рішення про фінансування даного проекту в 1,5 раза менше за нормоване ЧПД.

7. Якщо ризик є середнім, соціальне значення середнє, тоді міра прийняття рішення про фінансування проекту в 1,5 раза менше за нормоване ЧПД.

8. Якщо ризик є малим, соціальне значення середнє, тоді міра прийняття рішення про фінансування проекту менше ніж нормована міра ЧПД в три рази.

9. Якщо ризик є малим, а соціальне значення є високим, тоді міра прийняття рішення про фінансування проекту збігається з нормованою величиною ЧПД.

10. Якщо ризик є малим, а соціальне значення є нижче за середній рівень, тоді міра прийняття рішення про фінансування проекту менша ніж нормована величина ЧПД у разів п’ять.

11. Якщо ризик є середнім, соціальне значення і ЧПД проекту є малими, тоді міра прийняття рішення про фінансування проекту менша ніж нормована величина ЧПД у разів п’ять.

12. Якщо ризик є середнім, соціальне значення малим, тоді міра прийняття рішення про фінансування проекту менша ніж нормована величина ЧПД у разів п’ять.

13. Якщо ризик є середнім, соціальне значення посереднє, тоді міра прийняття рішення про фінансування проекту менша ніж нормована міра ЧПД у три рази.

14. Якщо ризик є середнім, а соціальне значення є високим, тоді міра прийняття рішення про фінансування проекту в 1,5 раза менша за нормовану величину ЧПД.

15. Якщо ризик є середнім, а соціальне значення є нижче за середній рівень, тоді міра прийняття рішення про фінансування проекту менша ніж нормована величина ЧПД у разів п’ять.

16. Якщо ризик є середнім, соціальне значення і ЧПД проекту є низькими, тоді приймається рішення про нефінансування даного проекту. Тобто можна вважати, що в такому «поганому» випадку ступінь фінансування проекту є 0 %.

17. Якщо ризик є високим, соціальне значення середнє, ЧПД велике, тоді міра прийняття рішення про фінансування проекту в 1,5 раза нижча нормованої величини ЧПД.

18. Якщо ризик є високим, соціальне значення посереднє, ЧПД середнє, тоді міра прийняття рішення про фінансування проекту менша ніж нормована міра ЧПД в три рази.

19. Якщо ризик є високим, а соціальне значення є високим, ЧПД високе, тоді міра прийняття рішення про фінансування проекту збігається з нормованою величиною ЧПД.

Для даного правила можна встановити ступінь істинності 0,75.

20. Якщо ризик є високим, а соціальне значення є нижче за середній рівень, ЧПД мале, тоді міра прийняття рішення про фінансування проекту 0 %.

3. Фазифікація вхідних і вихідних змінних

Як терм-множину першої вхідної змінної *«Ступінь ризику»* (Risk) будемо використовувати множину T1 = {*«допустимий»*, *«малий»*, *«середній»*, *«високий»*} або в символічному виді T1 = {RD, RM, RS, RV}.

Як терм-множину другої вхідної змінної *"Соціальне значення"* (Social-meaning) будемо використовувати аналогічну множину T2 = {*«низьке»*, *«нижче середнього»*, *«середнє»*, *«високе»*} або в символічному виді T2 = {SN, SNS, SS, SV}.

Як терм-множину третьої лінгвістичної змінної *«ЧПД»* (NPV) будемо використовувати множину T3={*«мале»*, *«середнє»*, *«високе»*} чи в символічному виді T3 = {CN, CS, CV}.

Як терм-множину вихідної лінгвістичної змінної *«Міра прийняття проекту на фінансування»"* (Credit) будемо використовувати множину T6 = {«0», «в 5 разів менша за ЧПД», «в 3 рази менша за ЧПД», «в 1,5 раза менша за ЧПД», «ЧПД», «1»} чи в символічному виді T6 = {P_0, P5, P3, P1_5, P1, P_1}.

4. Формування моделі системи нечіткого виводу

Наступним етапом побудови моделі є побудова бази правил. Для цієї мети будемо використовувати 20 правил нечітких продукцій, які зручно представити у вигляді наступної таблиці 3.3.1.

Таблиця 3.3.1

Правила нечітких продукцій для розглянутої системи нечіткого виводу

№ правила	Ступінь ризику	Міра соціального ефекту	ЧПД	Міра фінансування
1.	RD	SV	CV	P1
2.	RD	SS		P_1
3.	RM	SNS		P3
…				

Як схему нечіткого виводу, будемо використовувати метод Сугено. Далі необхідно визначити методи агрегування підумов. Оскільки у всіх правилах 1–40 як логічне зв'язування для підумов застосовується тільки нечітка кон'юнкція (операція «І»), то як метод агрегування будемо використовувати операцію min-кон'юнкції. Для акумуляції виводів правил будемо використовувати метод max-диз'юнкції, що також застосовується у випадку схеми нечіткого виводу методом Мамдані. Нарешті, як метод дефазифікації будемо використовувати метод центра ваги.

Розробку нечіткої моделі будемо виконувати з використанням графічних засобів системи MATLAB. З цією метою в редакторі FIS визначимо 3 вхідних змінних з іменами *«ступінь ризику»* (Risk), *«соціальне значення»* (Social), *«ЧПД»* (NPV) і одну вихідну змінну з ім'ям *«міра фінансування»* (Credit). Вид графічного інтерфейсу редактора FIS для цих змінних зображений на рис. 3.3.2.

Для вирішення поставленої задачі нечіткого моделювання будемо використовувати систему нечіткого виводу типу Сугено. Залишимо без зміни параметри розроблювальної нечіткої моделі, запропоновані системою MATLAB за замовчуванням.

Далі варто визначити функції приналежності термів для кожної з 3-х вхідних і єдиної вихідної змінних розглянутої системи нечіткого виводу. Для цієї мети скористаємося редактором функцій приналежності системи MATLAB. Графічний інтерфейс редактора функцій приналежності для вихідної змінної *«міра фінансування проекту»* зображений на рис. 3.3.3.

Далі задамо 20 правил для розроблювальної системи нечіткого виводу. Для цієї мети скористаємося редактором правил системи MATLAB. Вид графічного інтерфейсу редактора правил після задання всіх 20 правил нечіткого виводу зображений на рис. 3.3.4. Оскільки в робочому вікні відображаються не всі змінні нечіткої моделі, для управління режимом відображення змінних правил варто скористатися спеціальними кнопками, розташованими в нижній правій частині редактора правил.

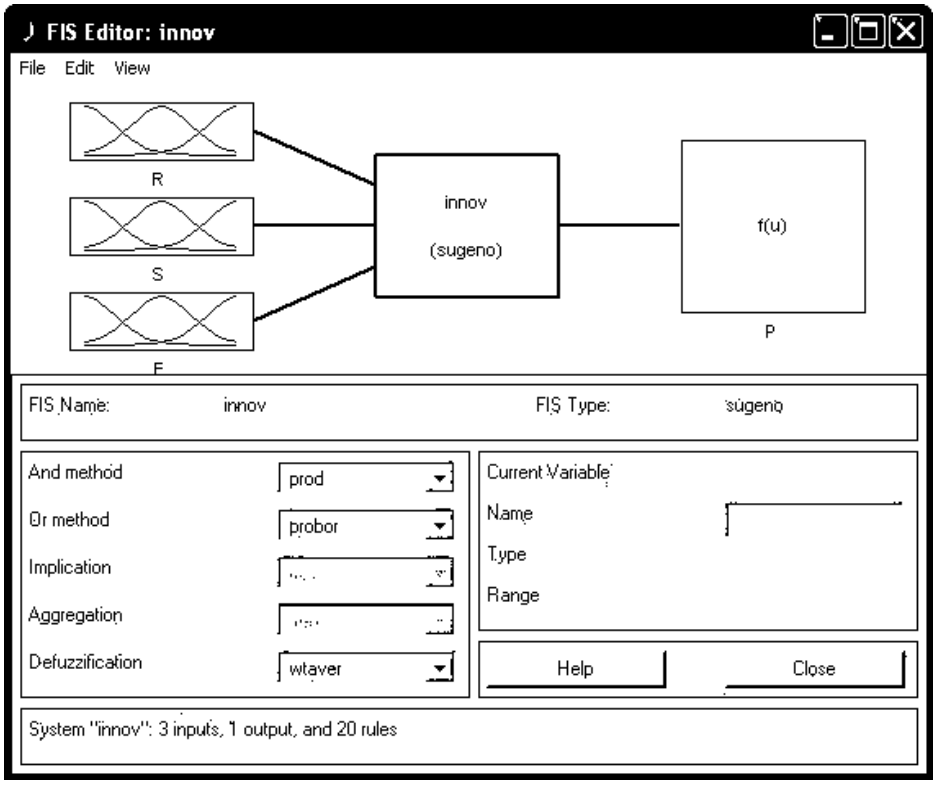

Рис. 3.3.2. Графічний інтерфейс редактора FIS після визначення вхідних і вихідної змінних системи нечіткого виводу mortgage

Тепер можна виконати аналіз побудованої системи нечіткого виводу для розглянутої задачі. З цією метою відкриємо вікно

перегляду правил системи MATLAB і введемо значення вхідних змінних для частинного випадку, коли значення вхідної змінної *«ступінь ризику»* оцінюється в 0.08, значення вхідної змінної *«соціальне значення»* також оцінюється в 0.48 балів, значення вхідної змінної *«ЧПД»* оцінюється в 900. Процедура нечіткого виводу, виконана системою MATLAB для розробленої нечіткої моделі, видає в результаті значення вихідної змінної *«міра фінансування проекту»*, рівне 50,6 % (рис. 3.3.5). Це досить висока оцінка фінансової заможності потенційного клієнта, що може служити підставою для позитивного рішення з боку банку про надання кредиту під заставу. Цей висновок цілком узгоджується з раніше висловленими інтуїтивними розуміннями.

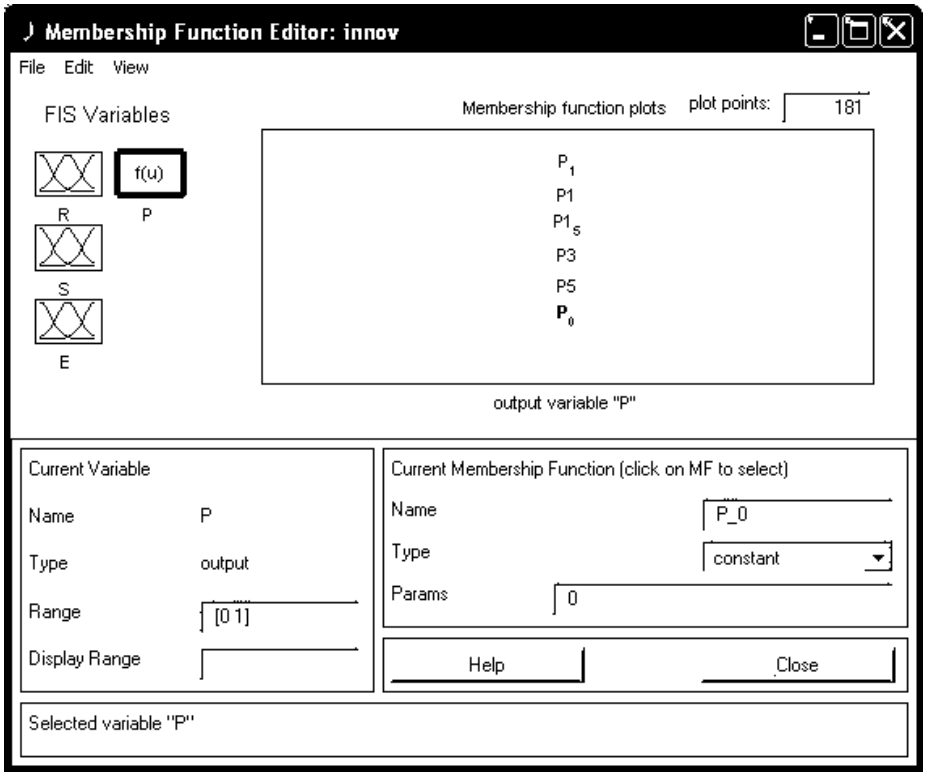

Рис. 3.3.3. Графічний інтерфейс редактора функцій приналежності для вихідної змінної *«міра фінансування»*

Далі можна визначити міру фінансування проекту для інших варіантів значень вхідних змінних і проглянути ті поверхні, які система пропонує як результат своєї роботи. Проаналізуйте їх (рис. 3.3.6).

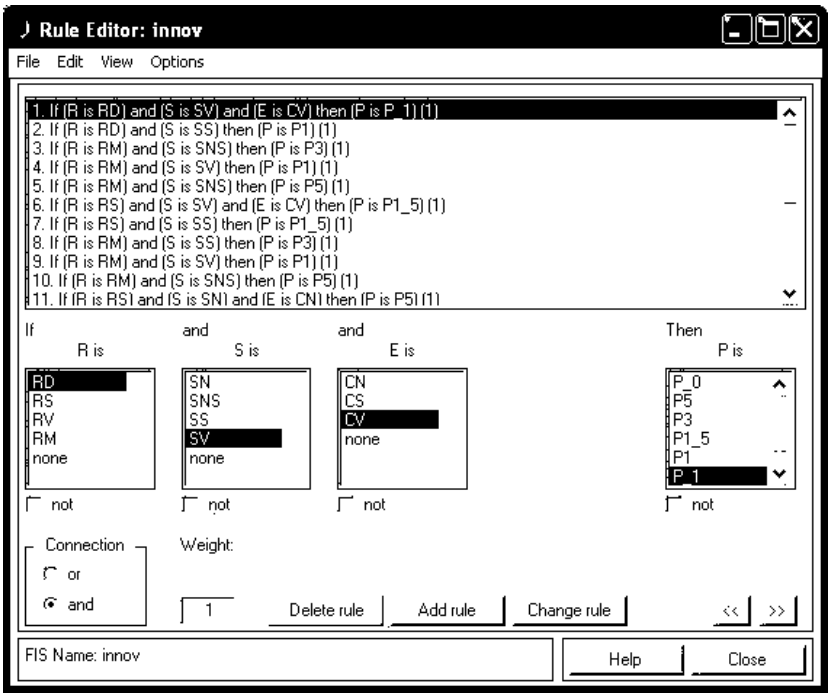

Рис. 3.3.4. Графічний інтерфейс введення правил

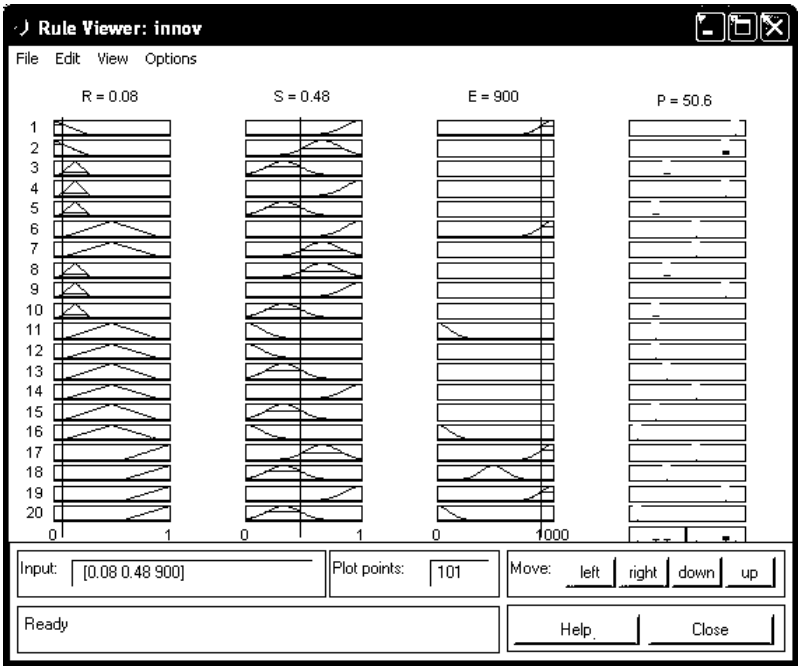

Рис. 3.3.5. Графічний інтерфейс програми перегляду правил після виконання процедури нечіткого виводу 2 варіанта

Ця поверхня нечіткого виводу дозволяє установити залежність значень вихідної змінної від значень окремих вхідних змінних нечіткої моделі. Аналіз цих залежностей може служити підставою для зміни функцій приналежності вхідних змінних чи нечітких правил з метою підвищення адекватності системи нечіткого виводу для конкретних стратегій інвесторів.

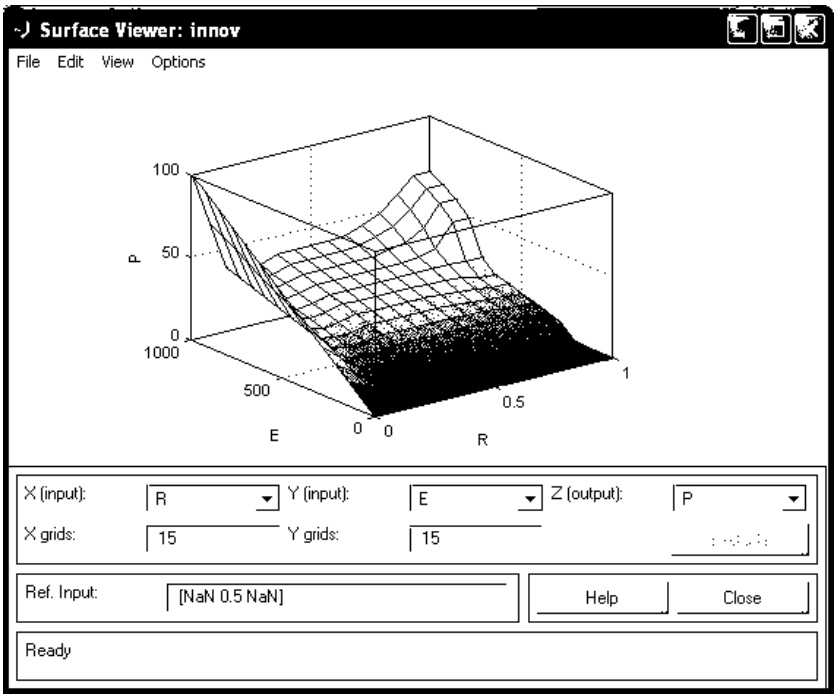

Рис. 3.3.6. Візуалізація поверхні нечіткого виводу розглянутої моделі

Слід зазначити також ту обставину, що розроблювачі даної нечіткої моделі відзначають її декілька спрощений характер порівняно з реально використовуваною у процесі прийняття рішень керівництвом інвестиційного фонду.

Завдання для самостійного виконання

Проектування системи оцінки фінансової заможності клієнтів

При видачі довгострокових кредитів на будівництво будинків чи котеджів під заставу нерухомості для оцінки заможності клієнтів банками традиційно використовується метод експертних оцінок. При цьому метою банків є одержання максимального прибутку від укладених угод з надання кредитів і виключення можливості фінансових утрат. Тому інтереси банків зосереджені, з одного боку, на збільшенні кількості успішних угод, а з іншого – на запобіганні невдалих угод, коли клієнт не повертає виданий кредит чи повертає його не вчасно.

Традиційно підставою для прийняття рішень з надання кредитів у майбутньому служить досвід успішних угод, зроблених у

минулому. Керівництво банку, в інтересах якого виконувалося відповідне дослідження, хотіло б узагальнити правила надання кредитів з метою максимально повно використовувати досвід експертів. При цьому необхідно виключити можливі помилки суб'єктивного характеру з боку окремих менеджерів у випадку неадекватного оцінювання фінансової заможності клієнтів.

Аналіз стратегії надання кредитів на будівництво будинків показує, що для оцінювання фінансової заможності клієнтів можуть бути використані різні характеристики, такі як місцезнаходження споруджуваного будинку, якість передбачуваного виконання опоряджувальних робіт, оцінка активів потенційного клієнта, оцінка доходу потенційного клієнта за винятком фіксованих витрат, величина відсотків з кредиту, що підлягають сплаті. При цьому власне фінансова заможність клієнта оцінюється його кредитоспроможністю.

Однією з перших формальних моделей, запропонованих для розв'язку даної задачі, була статистична модель, заснована на імовірнісній інтерпретації кількісної оцінки позитивного рішення про надання кредиту. Однак більш детальний аналіз цієї моделі згодом показав її неадекватність, пов'язану з недостатнім обсягом статистичної вибірки й умовами надання кредитів, що змінюються з плином часу.

Як перша вхідна змінна, використовується оцінка місця розташування споруджуваного будинку, що безпосередньо оцінює проект споруджуваного будинку, беручи до уваги розміщення будинку в тому чи іншому конкретному районі міста чи регіоні передмістя. Очевидно, чим вище ця оцінка, тим більше ліквідним є проект у випадку його реалізації на ринку нерухомості.

У ролі другої вхідної змінної використовується якість передбачуваного виконання опоряджувальних робіт відповідно до архітектурного проекту споруджуваного будинку. Ця змінна вносить додатковий елемент в оцінку вартості споруджуваного будинку.

Як третю вхідну змінну використовують оцінку активів для оцінки майна чи авуарів у випадку неспроможності потенційного

клієнта при неповерненні їм узятого кредиту. Дійсно, величина наданого кредиту повинна ґрунтуватися не тільки на урахуванні вартості споруджуваного будинку, але і на власній капіталізації клієнта.

Четвертою вхідною змінною використовується оцінка доходу потенційного клієнта за винятком фіксованих витрат, яка також використовується у випадку неспроможності потенційного клієнта при неповерненні їм узятого кредиту. Чим вище значення цієї змінної, тим більш успішним є надання кредиту клієнту.

П’ятою вхідною змінною використовується величина відсотків, що підлягають сплаті, відповідно до передбачуваного плану виплат з узятого кредиту. Ця змінна пов’язана з терміном надання кредиту і його величиною, дозволяючи об’єднати в собі відповідні характеристики кредиту. Чим вище величина виплат по відсотках, тим більше високими повинні бути значення активів і доходів для позитивного рішення про надання кредиту потенційному клієнту.

Як **вихідна** змінна використовується **оцінка кредитоспроможності**, що є основою для ухвалення рішення керівництвом банку з надання кредиту потенційним клієнтам. При цьому рішення про надання кредиту керівництвом банку приймається тільки у випадку високої оцінки цієї вихідної змінної.

Аналіз надання кредитів на будівництво будинків показує, що для аналізу фінансової заможності потенційних клієнтів керівництво банків застосовує певні евристичні правила.

Після розгляду змістовної постановки задачі можна приступити до побудови її нечіткої моделі у формі відповідної системи нечіткого виводу. Для цієї мети скористаємося розглянутими раніше графічними засобами пакета Fuzzy Logic Toolbox системи MATLAB.

При побудові нечіткої моделі оцінки фінансової заможності потенційних клієнтів було зроблено припущення про те, що всі розглянуті змінні вимірюються в балах в інтервалі дійсних чисел від 0 до 10. При цьому найнижча оцінка значення кожної із змінних є 0, а найвища – 10.

Як терм-множину першої вхідної змінної *«Місцезнаходження»* (Location) будемо використовувати множину T1 = {*«непрестижне», «престижне», «дуже престижне»*} або в символічному виді T1 = {PS, PM, PB}.

Як терм-множину другої вхідної змінної *«Опоряджування»* (Work-manship) будемо використовувати аналогічну множину T2 = {*«погане», «гарне», «прекрасне»*} або в символічному виді T2 = {PS, PM, PB}.

Як терм-множину третьої лінгвістичної змінної *«Активи»* (Asset) будемо використовувати множину T3 = {*«низькі», «середні», «високі»*} чи в символічному виді T3 = {PS, PM, PB}.

Як терм-множину четвертої лінгвістичної змінної *«Дохід»* (Income) будемо використовувати аналогічну множину T4 = {*«низький», «середній», «високий»*} чи в символічному виді T4 = {PS, PM, PB}.

Як терм-множину п’ятої лінгвістичної змінної *«Виплати»* (Interest) будемо використовувати аналогічну множину T5 = {*«низькі», «середні», «високі»*} чи в символічному виді T5 = {PS, PM, PB}.

Як терм-множину вихідної лінгвістичної змінної *«Кредитоспроможність»* (Credit) будемо використовувати множину T6 = {*«дуже низька», «низька», «середня», «висока», «дуже висока»*} чи в символічному виді T6 = {NB, NS, Z, PS, PB}.

Як схему нечіткого виводу, будемо використовувати метод Мамдані, тому методом активації буде MIN. Далі необхідно визначити методи агрегування підумов. Оскільки у всіх правилах 1–40 як логічне зв’язування для підумов застосовується тільки нечітка кон’юнкція (операція «І»), то як метод агрегування будемо використовувати операцію min-кон’юнкції. Для акумуляції виводів правил будемо використовувати метод max-диз’юнкції, що також застосовується у випадку схеми нечіткого виводу методом Мамдані. Нарешті, як метод дефазифікації будемо використовувати метод центра ваги.

Розробку нечіткої моделі (назвемо її mortgage) будемо виконувати з використанням графічних засобів системи MATLAB. Із цією метою в редакторі FIS визначимо 5 вхідних змінних з іме-

нами *«місцезнаходження»* (β1), *«опоряджування»* (β2), *«активи»* (β3), *«дохід»* (β4), *«виплати»* (β5) і одну вихідну змінну з ім'ям *"кредитоспроможність"* (β6). Для вирішення поставленої задачі нечіткого моделювання будемо використовувати систему нечіткого виводу типу Мамдані. Залишимо без зміни параметри розроблювальної нечіткої моделі, запропоновані системою MATLAB за замовчуванням, а саме, логічні операції (**min** – для нечіткого логічного І, **max** – для нечіткого логічного АБО), метод імплікації (**min**), метод агрегування (**max**) і метод дефазифікації (**centroid**).

Слід зазначити також ту обставину, що розроблювачі даної нечіткої моделі відзначають її декілька спрощений характер порівняно з реально використовуваною у процесі прийняття рішень керівництвом банку. Водночас розглянута нечітка модель має досить високу адекватність, що обумовлює її успішне застосування у практиці фінансових операцій деяких банків.

Таблиця 3.3.2

Встановлення ваги правил

№ правила	**Вага**	**№ правила**	**Вага**
1	0.51	11	0.28
2	0.75	12	0.97
3	0.24	13	0.32
4	0.14	14	0.71
5	0.77	15	0.47
6	0.33	16	0.69
7	0.79	17	0.96
8	0.16	18	0.74
9	0.86	19	0.86
10	0.98	20	0.11
№ правила	**Вага**	**№ правила**	**Вага**
21	0.53	31	0.05
22	0.09	32	0.03
23	0.81	33	0.49
24	0.70	34	0.68
25	0.06	35	0.27
26	0.58	36	0.40
27	0.78	37	0.59
28	0.91	38	0.14

29	0.17	39	0.75
30	0.34	40	0.59

ЛАБОРАТОРНА РОБОТА 3.4

Тема. Розробка систем інтелектуального аналізу даних методами нечіткої кластеризації

Мета роботи: засвоїти методику проектування і побудови систем інтелектуального аналізу даних на основі методів нечіткої кластеризації.

Теоретичні відомості

Кластерний аналіз – це сукупність методів, підходів і процедур, які розробляються для розв’язування проблеми формування класів – сукупностей даних, однорідних за заданими ознаками.

Кластерний аналіз (автоматична класифікація сукупності даних) займає одне із центральних місць серед методів аналізу даних і являє собою сукупність підходів та алгоритмів знаходження деякого розбиття досліджуваної сукупності об’єктів на підмножини щодо схожих між собою елементів. Такі підмножини отримали назву кластерів.

Виділення кластерів серед сукупності даних повинне відповідати таким вимогам:

1) кожний кластер являє собою сукупність об’єктів, які схожі між собою значеннями деяких властивостей або ознак;

2) сукупність усіх кластерів має бути вичерпаною, тобто всі об’єкти досліджуваної сукупності повинні належати до деякого кластеру;

3) кластери мають бути взаємовиключні; тобто жоден з об’єктів не повинен належати до двох різних кластерів.

Формально, під задачею кластерного аналізу розуміється задача знаходження деякого теоретико-множинного розбиття початкової множини об’єктів на підмножини, які не перетинаються, таким чином, щоб елементи, які належать до однієї підмножини,

відрізнялися між собою значно меншою мірою, ніж об'єкти з різних підмножин.

Концептуальний зв'язок між кластерним аналізом і теорією нечітких множин оснований на тому, що при розв'язуванні задач структуризації складних систем більшість класів, що формуються, «розмиті» за своєю природою. Тому найбільш адекватну відповідь слід шукати не на питання «Чи належить елемент до того чи іншого класу?», а на питання «Якою мірою даний елемент належить класу, що розглядається?».

Методи нечіткої кластеризації вводять до розгляду нечіткі кластери і відповідні їм функції приналежності, які приймають значення з інтервалу $[0,1]$.

Таким чином, задача нечіткої кластеризації полягає у тому, що необхідно знайти нечітке розбиття або нечітке покриття множини елементів сукупності, що досліджується. Задача зводиться до знаходження міри приналежності елементів множини нечітким кластерам (класам).

Нехай початкова (досліджувана) сукупність даних являє собою скінченну множину елементів $A = \{a_1, a_2, ..., a_n\}$, яку ще називають множиною об'єктів кластеризації. Вводиться також скінченна множина ознак або атрибутів об'єктів $P = \{p_1, p_2, ..., p_q\}$, кожний з яких являє собою деяку характеристику елементів множини A.

Далі пропонується, що для всіх елементів множини об'єктів кластеризації виміряли всі ознаки множини P, і кожен елемент множини $a_i \in A$ представлений вектором $x_i = \{x_i^1, x_i^2, ..., x_i^q\}$, де $x_i^j \in R$ – дійсне значення ознаки $p_j \in P$ для об'єкта $a_i \in A$.

Проблема кількісного вимірювання ознак кожного об'єкта з сукупності – нетривіальна і самостійна задача. Процес вимірювання ознак може бути реалізований у різних шкалах, кожна з яких характеризується допустимим перетворенням даних. У зв'язку з цим визначають різні типи шкал:

– шкала найменувань: об'єкту ставиться у відповідність деякий символ або номер, який лише відокремлює одне значення ознаки від іншого; прикладом таких ознак є стать людини – (м, ж)

або найменування міст (Київ, Житомир, Луганськ,...); допустимим відображенням множини об’єктів у множину символів є бієктивне відображення;

– порядкова шкала: разом з відповідною множиною символьних ознак об’єктів ця шкала дозволяє встановити відношення порядку щодо цієї ознаки; тоді об’єкту ставиться у відповідність деяке число, яке відіграє роль його оцінки в балах; допустимим відображенням є монотонне зростаюче відображення або функція між двома множинами значень ознак; приклад – оцінки на іспитах;

– інтервальна шкала: крім порядку елементів за ознаками, ця шкала встановлює рівність інтервалів значень цієї ознаки; об’єкту, як правило, ставиться у відповідність число, яке дорівнює значенню цієї ознаки; допустимим перетворенням тут є довільна лінійна зростаюча функція між двома множинами значень ознак; характерною ознакою такої шкали є відсутність абсолютного нуля; приклад – температура в шкалах Цельсія;

– шкала відношень: у доповнення до рівності інтервалів додає ще рівність відношень значень ознаки, що розглядається; об’єкту ставиться у відповідність деяке число, яке дорівнює значенню цієї ознаки; допустимим відображенням є довільна лінійна зростаюча функція, яка проходить через нуль; приклад – відстань у метрах, швидкість у км/ч.

Множину ознак слід обирати таким чином, щоб всі $x_i^j \in R$ були виміряні в шкалах відношень чи інтервалів. Саме в такому випадку результати нечіткої кластеризації мають змістовну інтерпретацію, яка адекватна проблемі знаходження нечітких кластерів.

Вектори значень ознак $x_i = \left\{x_i^1, x_i^2, ..., x_i^q\right\}$ зручно представляти у вигляді матриці даних D розмірності (n*q), кожний рядок якої є елементами вектора x_i .

Отже, **задача нечіткого кластерного аналізу формулюється таким чином**: на основі даних матриці D визначити таке нечітке розбиття $R(A) = \left\{A_k \mid A_k \subseteq A\right\}$ або нечітке покриття

$J(A) = \{A_k \mid A_k \subseteq A\}$ множини А на задане число нечітких кластерів $A_k (k \in \{2,...,c\})$, яке доставляє екстремум деякій цільовій функції $F(R(A))$ серед усіх нечітких розбиттів чи екстремум цільової функції $F(J(A))$ серед усіх можливих нечітких покриттів.

Для конкретизації задачі ще слід уточнити вигляд цільової функції та тип шуканих нечітких кластерів.

Одним з видів конкретизації цієї задачі є використання спеціальної функції fcm системи MATLAB, який оснований на алгоритмі розв'язування методом нечітких с-середніх.

Для уточнення вигляду цільової функції $F(J(A))$ вводяться деякі додаткові поняття. По-перше, пропонується, що шукані нечіткі кластери являють собою нечіткі множини $A_k (k \in \{2,...,c\})$, які є нечітким покриттям початкової множини об'єктів кластеризації А, для якої має місце така умова:

$$\sum_{k=1}^{c} \mu_{A_k}(a_i) = 1, (\forall a_i \in A), \qquad (3.1)$$

де с – загальна кількість нечітких кластерів $A_k (k \in \{2,...,c\})$, яке вважається попередньо заданим.

Далі для кожного кластеру вводяться так звані типові представники або **центри** v_k шуканих нечітких кластерів $A_k (k \in \{2,...,c\})$, які розраховуються за такою формулою:

$$v_j^k = \frac{\sum_{i=1}^{n} (\mu_{A_k}(a_i))^m x_i^j}{\sum_{i=1}^{n} (\mu_{A_k}(a_i))^m}, (\forall k \in \{2,...,c\}, \forall p_j \in P), \qquad (3.2)$$

де m – деякий параметр, який має назву **експоненційна вага** і дорівнює деякому дійсному числу (m>1). Кожний із центрів кластерів є вектором $v_k = (v_k^1, v_k^2, ..., v_k^q)$ у деякому q-вимірному нормованому просторі ознак, який ізоморфний R^q, якщо всі ознаки виміряні за шкалою відношень.

Як цільову функцію, розглядатимемо суму квадратів зважених відхилень координат об'єктів кластеризації від центрів нечітких кластерів:

$$F(A_k, v_k^j) = \sum_{i=1}^{n} \sum_{k=1}^{c} (\mu_{A_k}(a_i))^m \sum_{j=1}^{q} (x_i^j - v_k^j)^2 . \qquad (3.3)$$

Чим більше елементів містить множина А, тим менше значення слід вибирати для m > 1.

Тоді задача нечіткої кластеризації полягає у наступному: для заданої матриці даних D, кількості нечітких кластерів $c \in N, c > 1$ **, параметра m, визначити матрицю U значень функції приналежності об'єктів кластеризації** $a_i \in A$ нечітким кластерам $A_k (k \in \{2,...,c\})$, які доставляють мінімум цільової функції (4.3) і задовольняють обмеженням (4.1)–(4.2), а також додатковим обмеженням:

$$\begin{aligned} &\sum_{i=1}^{n} \mu_{A_k}(a_i) > 0, (\forall k = \{2,...,c\}) \\ &\mu_{A_k}(a_i) \geq 0 (\forall k = \{2,...,c\}, a_i \in A) \end{aligned} \qquad (3.4)$$

Умови (4.4) виключають появу пустих нечітких кластерів у шуканій нечіткій кластеризації. Таким чином, мінімізація цільової функції (4.3) – це мінімізація відхилення всіх об'єктів кластеризації від центрів нечітких кластерів пропорційно значенням функцій приналежності цих об'єктів відповідним нечітким кластерам.

Ця функція не є опуклою, а тому задача кластеризації в загальному випадку належить до багатокритеріальних задач нелінійного програмування.

Алгоритм розв'язування задачі нечіткої кластеризації

Основні ідеї алгоритму для розв'язування задачі нечіткої кластеризації були запропоновані Дж. К. Данном у 1974 р. Цей алгоритм спочатку отримав назву нечіткого алгоритму fuzzyISODATA. У 1980 році Дж. К. Беджек теоретично довів збіжність цього алгоритму, потім він же узагальнив цей алгоритм на випадок довільних нечітких множин даних і запропонував для цього алгоритму назву нечітких середніх FCM, Fuzzy-C-Means. Саме під такою назвою алгоритм реалізований у системі MATLAB.

Алгоритм FCM має ітеративний характер послідовного покращення деякого початкового нечіткого розбиття $R(A) = \{A_k \mid A_k \subseteq A\}$, яке задається користувачем або формується автоматично за деяким евристичним правилом. На кожному кроці ітерації прораховуються значення функцій приналежності нечітких кластерів і їх типових представників.

Алгоритм FCM завершує роботу у випадку, коли відбудеться наперед задане число ітерацій, або коли мінімальна абсолютна різниця між значеннями функцій приналежності на двох послідовних ітераціях не стане менше деякого наперед заданого значення.

Формально алгоритм FCM представляється у вигляді таких кроків:

1. Попередньо необхідно задати такі значення: кількість шуканих нечітких кластерів c, максимальну кількість ітерацій алгоритмів $s \in N$, параметр збіжності алгоритму $\varepsilon \in R_+$, а також експоненційну вагу для цільової функції і центрів кластерів m>1. Як початкове розбиття на першій ітерації алгоритму для матриці даних D задати деяке нечітке розбиття $R(A) = \{A_k \mid A_k \subseteq A\}$ на c непустих нечітких кластерів, які описуються сукупністю функцій приналежності $\mu_k(a_i), \forall k \in \{2,...,c\}, \forall a_i \in A$.

2. Для поточного нечіткого розбиття $R(A) = \{A_k \mid A_k \subseteq A\}$ за формулою (6.2) розрахувати центри нечітких кластерів $v_k^j, (\forall k = \{2,...,c\}, \forall p_j \in P$ і значення цільової функції (6.3). Кількість виконаних ітерацій покласти 1.

3. Сформувати нове нечітке розбиття $R'(A) = \{A_k \mid A_k \subseteq A\}$ множини об’єктів кластеризації A на c непусті нечіткі кластери, які характеризуються сукупністю функцій приналежності $\mu_k^1(a_i), \forall k \in \{2,...,c\}, \forall a_i \in A$, що визначаються за формулою:

$$\mu_k^1(a_i)=\left(\sum_{l=1}^{c}\left(\frac{\left(\sum_{j=1}^{q}(x_i^j-v_k^j)^2\right)^{\frac{1}{2}}}{\left(\sum_{j=1}^{q}(x_i^j-v_l^j)^2\right)^{\frac{1}{2}}}\right)^{\frac{2}{m-1}}\right)^{-1},\forall k\in\{2,...,c\},\forall a_i\in A$$

4. При цьому, якщо для деякого $k\in\{2,...,c\}$ і деякого $a_i\in A$ значення $\sum_{j=1}^{q}(x_i^j-v_k^j)^2=0$, тоді для відповідного нечіткого кластеру $\mathrm{A_k}$ беремо $\mu_k^1(a_i)=1$, а для інших кластерів $A_l(\forall l=\{2,...,c\},l\neq k)$ беремо $\mu_l^1(a_i)=0$. Якщо ж таких значень $k\in\{2,...,c\}$ виявиться декілька для $a_i\in A$, тоді евристично беремо $\mu_{\mathrm{k}}^1(\mathrm{a_i})=1$ для меншого з них, а для інших $\mu_l^1(a_i)=0$.

5. Для нового нечіткого розбиття $R'(A)=\{A_k\mid A_k\subseteq A\}$ за формулою (6.2) розраховуємо центри нечітких кластерів і значення цільової функції (4.3).

6. Якщо кількість виконаних ітерацій більше за s або модуль різниці між попереднім і новим значенням цільової функції менше за $\varepsilon\in R_+$, тоді як результат прийняти нечітке розбиття $R'(A)=\{A_k\mid A_k\subseteq A\}$ і завершити виконання алгоритму. Інакше, вважати поточним розбиттям $R(A)=R'(A)$ і перейти на крок 3, збільшивши на 1 кількість виконаних ітерацій.

У результаті виконання алгоритм зводиться до деякого локально-оптимального розбиття $R^*(A)$, яке описується сукупністю функцій приналежності $\mu_k(a_i)$, а також центрами нечітких класів $v_k=(v_k^1,v_k^2,...,v_k^q)$.

Виконання алгоритму FCM у системі MATLAB

Функція fcm може бути викликана в одному з наступних форматів:

$$[center,U,obj\_fcn]=fcm(data,cluster\_n)$$

або

$$[center, U, obj\_fcn] = fcm(data, cluster\_n, options).$$

Вхідними аргументами цієї функції є:

– data: матриця початкових даних D, і-тий рядок якої являє собою інформацію про об'єкт нечіткої кластеризації $a_i \in A$ у формі вектора $x_i = \{x_i^1, x_i^2, ..., x_i^q\}$;

– cluster_n: число шуканих кластерів $c \in N, c > 1$.

Вихідними аргументами цієї функції є:

– center: матриця центрів шуканих нечітких кластерів, кожний рядок якої являє собою координати центру одного з нечітких кластерів у формі вектора $v_k = (v_k^1, v_k^2, ..., v_k^q)$;

– U: матриця значень функцій приналежності шуканого нечіткого розбиття $\mu_k(a_i), \forall k \in \{2, ..., c\}, \forall a_i \in A$;

– obj_fun: значення цільової функції (3) на кожній з ітерацій роботи алгоритму.

Функція fcm() може бути викликана з додатковими аргументами options, які введенні для управління процесом кластеризації, а також для зміни критерію роботи алгоритму і/або відображення інформації на екрані монітора.

Ці додаткові аргументи мають такі значення:

– option (1): експоненційна вага m для розрахунків матриці нечіткого розбиття U (за замовченням $m = 2$);

– option (2): максимальне число ітерацій s (за замовченням це значення дорівнює 100);

– option (3): параметр збіжності алгоритму ε (за замовченням це значення дорівнює 0.00001);

– option (4): інформація про поточну ітерацію, яка відображається на екрані монітора (за замовченням, це значення 1).

Якщо будь-яке зі значень додаткових аргументів дорівнює NaN (не число), тоді для цього аргументу використовується значення за замовченням.

Приклад реалізації алгоритму

Завдання 1. Як приклад застосування нечіткої кластеризації, розглянемо множину даних, які містяться в системі MATLAB

і використовуються як текстова сукупність об’єктів нечіткої кластеризації. Ці дані являють собою матрицю D розмірності 140x2 і містяться у файлі fcmdata.dat, який поставляється разом зі MATLAB. У цьому випадку матриця D відповідає 140 об’єктам, для кожного з якого виконане вимірювання за двома ознаками, що є дуже зручним для візуалізації результатів нечіткої кластеризації у двовимірному просторі на площині.

1. Для візуалізації цих даних слід виконати такі команди:

Load fcmdata.dat

plot(fcmdata(:,1), fcmdata(:,2), ‘o’)

На екрані з’явиться графічне зображення, яке представлене на рис. 3.4.1.

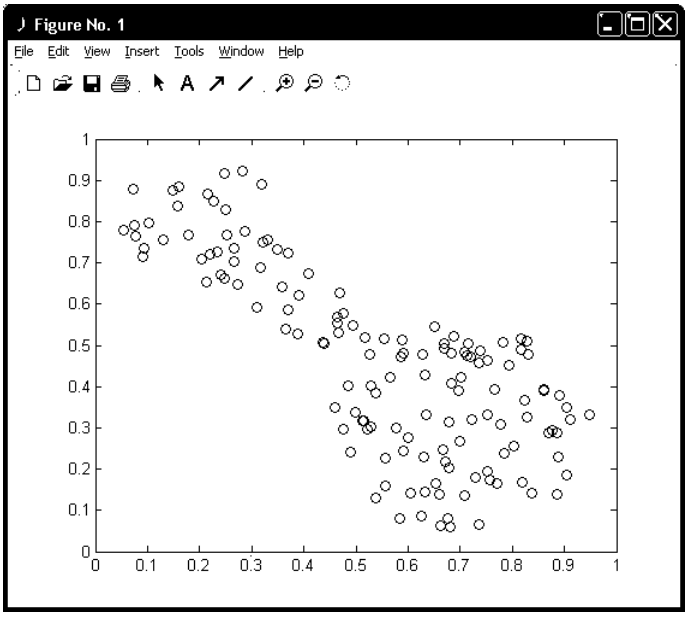

Рис. 3.4.1. Зображення точок матриці D з файлу fcmdata.dat

2. Далі слід викликати функцію fcm, наприклад, з таким форматом:

[center, U, obj_fcn]=fcm(fcmdata, 2).

Потім слід подивитись результати виконання процедури нечіткої кластеризації:

– координати центрів класів, тобто, матрицю center;

– належність кожної сукупності даних до класів – матрицю U;

– значення функції цілі – obj_fcn.

Взагалі процедуру нечіткої кластеризації можна записати у вигляді командного М-файла з таким текстом:

```
load fcmdata.dat
plot(fcmdata(:,1), fcmdata(:,2),’o’)
[center, U, obj_fcn]= fcm(fcmdata, 2);
maxU=max(U);
index1 = find(U(1,:)== maxU);
index2 = find(U(2,:)== maxU);
line(fcmdata(index1,1), fcmdata(index1,2),’linestyle’,’none’,…
‘marker’, ‘x’, ‘color’, ‘g’);
line(fcmdata(index2,1), fcmdata(index2,2),’linestyle’,’none’,…
‘marker’, ‘x’, ‘color’, ‘r’);
hold on
plot( center(1,1), center(1,2),’ko’, ‘markersize’,10, ‘LineWidth’, 2)
plot( center(2,1), center(2,2),’ko’, ‘markersize’,10, ‘LineWidth’, 2)
```

Результатом цієї програми буде розбиття даних на два кластери, візуалізація якого зображена на рис. 3.4.2.

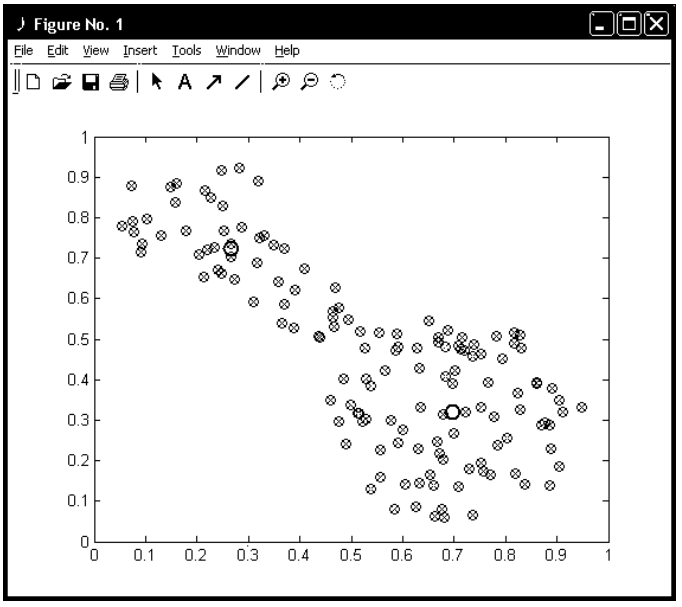

Рис. 3.4.2. Результат роботи програми нечіткої кластеризації

3. Після роботи програми в системі MATLAB можна перевірити значення матриць center та U, набравши їх назву в командному рядку і натиснувши Enter.

4. Крім того, у програмі можна використати такий формат запису:

[center, U, obj_fcn] = fcm(fcmdata, 2, [2.5 1000 0.000001 1]);

У цьому випадку експоненційна вага 2.5, максимальне число ітерацій 1000, параметр збіжності $\varepsilon = 0.000001$. Порівняльний аналіз показує практичну ідентичність графіків – результатів використання обох форматів функції fcm, що дозволяє зробити висновок про відповідність отриманих результатів нечіткої кластеризації.

5. Для розв'язування задачі нечіткої кластеризації в системі MATLAB можна використовувати графічний інтерфейс, який викликається за допомогою команди **findcluster**. Ця програма може використовувати або метод с-середніх або метод субтрактивної кластеризації (substractive clustering, який викликається і окремо за допомогою команди **subclust**). Останній використовується тоді, коли не можна заздалегідь встановити число кластерів с на кроці 6.

Формат виклику графічного інтерфейсу: **findcluster або findcluster('file.dat')**.

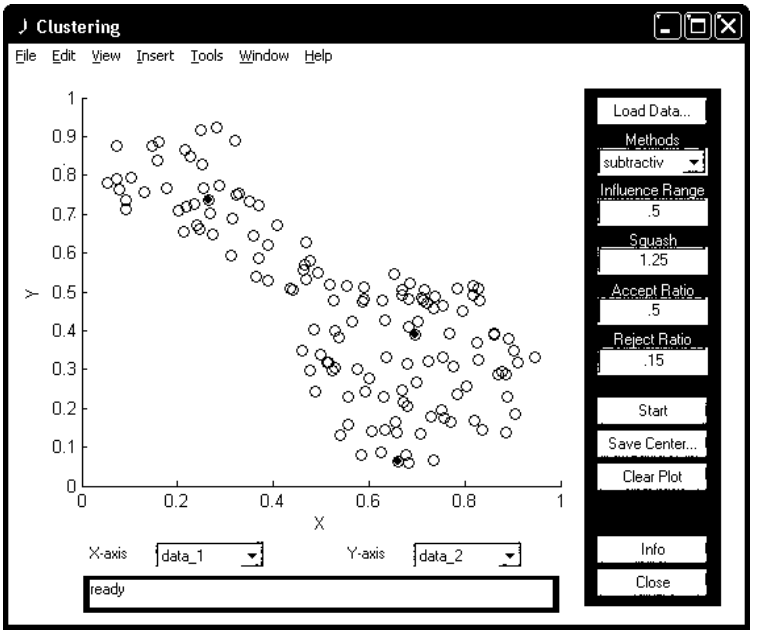

Рис. 3.4.3. Вікно роботи графічного інтерфейсу нечіткої кластеризації для алгоритму субтрактивної кластеризації

У цьому вікні можна завантажити файл даних Load Data, обрати метод кластеризації Methods, обрати необхідні значення параметрів і натиснути кнопку Start.

6. Нехай заздалегідь невідома кількість кластерів с. Тоді слід використати метод субтрактивної кластеризації. Ідея цього методу полягає у тому, що кожна точка даних пропонується як центр потенційного кластеру. Далі слід вирахувати деяку міру можливості кожної точки даних представляти центр кластеру. Ця кількісна міра основана на оцінці густини точок навколо відповідного центра кластера.

Цей алгоритм, який є узагальненням методу кластеризації Р. Ягера, заснований на виконанні таких кроків:

1) вибрати точку даних з максимальним потенціалом для представлення центру першого кластеру;

2) забрати всі точки даних в околі центру даного кластеру, величина якої задається параметром **radii**, щоб визначити такий нечіткий кластер і координати його центру.

Далі ці дві процедури повторюються до тих пір, доки всі точки даних не лежатимуть у границях околів радіуса **radii** навколо шуканих кластерів.

Функція командного рядка

[C, S] = subclust (X, radii, xBounds, options)

знаходить центри таких кластерів.

При цьому матриця X містить об’єкти кластеризації, кожний рядок якої відповідає координатам окремої точки даних. Параметр radii являє собою вектор, компоненти якого приймають значення з інтервалу [0,1] і задають діапазон розрахунку центрів кластерів по кожній з вимірювань об’єктів. Робиться припущення, що всі дані перебувають у деякому гіперкубі. Взагалі, малі значення параметрів radii призводять до знаходження малого числа великих за кількістю точок кластерів. Найкращих результатів можна очікувати при значенні radii між 0.2 і 0.5.

Аргумент xBounds являє собою матрицю розміру (2xq), яка визначає засіб відображення матриці даних X у деякому одиничному гіперкубі. Тут q – кількість ознак. Цей аргумент є необов’язковим, якщо матриця X вже нормована. Перший рядок цієї

матриці містить мінімальні значення інтервалу вимірювання кожної ознаки, а другий рядок – максимальне значення вимірювання кожної ознаки.

Для зміни значень, які встановлені по замовченню, можна використати параметр options, компоненти вектора якого можуть приймати такі значення:

– options(1) = guashFactor – параметр, який використовується як коефіцієнт для множення значень radii з ціллю зменшення впливу потенціалу граничних точок, які розглядаються як частина даного кластеру (за замовченням це значення дорівнює 1.25);

– options(2) = acceptRatio – параметр, який встановлює потенціал як частину потенціалу першого кластеру, вище якого інша точка даних не може розглядатися як центр іншого кластеру (за замовченням це значення 0.5);

– option(3) = rejectRation – параметр, який встановлює потенціал як частину потенціалу першого кластеру, нижче якого інша точка даних не може розглядатися як центр іншого кластеру (за замовченням це значення 0.15);

– options(4) = verbose – якщо значення цього параметра не дорівнює 0, тоді на екран монітору виводиться інформація про виконання процесу кластеризації (за замовченням це значення 0).

Функція subclust повертає матрицю C значень координат центрів нечітких кластерів. При цьому кожний рядок цієї матриці містить координати одного центру кластеру. Вектор S містить σ-значень, які визначають діапазон впливу центра кластеру за кожною з розглянутих ознак. При цьому всі центри кластерів мають однакову множину σ-значень.

Для **прикладу** розглянемо таку послідовність команд:

Load fcmdata.dat

[C, S] = subclust(fcmdata, [0.5 0.5], [], [1.25 0.5 0.15 1])

Для кожної ознаки вводяться радіуси околів – 0.5 і 0.5.

Як видно з рис. 4.3, для цієї сукупності даних функція suclust має знайти три кластери.

Для розв’язування задачі субтрактивної кластеризації може використовуватися і розглянутий раніше графічний інтерфейс, який викликається командою **findcluster**.

Таким чином, система MATLAB дозволяє розв’язувати задачі нечіткої кластеризації двома способами: за допомогою функцій командного рядка і за допомогою графічного інтерфейсу кластеризації. Результати нечіткої кластеризації мають наближений характер і мають служити для попередньої структуризації вхідної (початкової) інформації про систему, що вивчається. Тобто, провівши кластеризацію, можна отримати і зберегти знання про систему у вигляді структуризації початкової інформації.

Завдання для самостійної роботи

1. Виконати нечітку кластеризацію набору точок у двовимірному просторі ознак: (1,2); (2,1); (1, 2); (3,2); (2,3); (1,6); (2,5); (3,6); (5,5); (5,6); (7,5); (7,6). Розділити їх на 3 кластери. Відповідь можна подивитися на рис. 3.4.4.

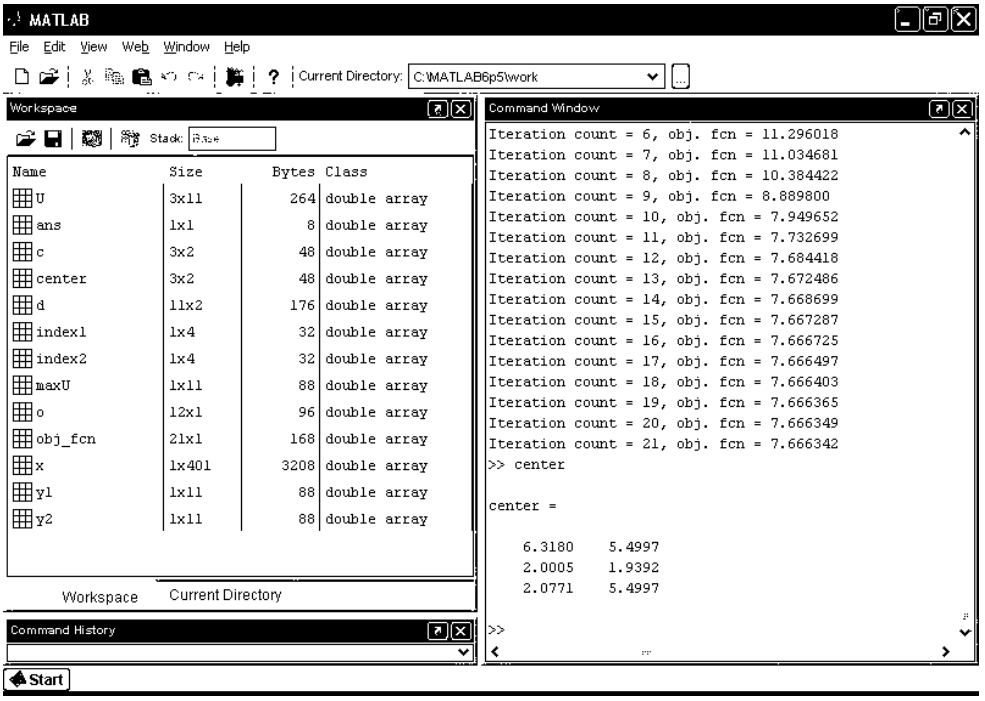

Рис. 3.4.4. Точки і центри кластерів завдання 1

2. Виконати нечітку кластеризацію 10 фірм, які можуть у майбутньому опинитися в різних зонах своєї діяльності на ринку (2, 3 або 4 зони). Для цього слід використовувати дві ознаки ста-

ну фірм – середній дохід і середньо-квадратичне відхилення можливого доходу, тобто ризикованість. Відомі такі дані про фірми:

Фірма	**Показники**					
1	Можливий дохід	100	120	300	250	200
	Імовірність	0,1	0,3	0,2	0,2	0,2
2	Можливий дохід	120	400	200	150	300
	Імовірність	0,2	0,4	0,1	0,2	0,1
3	Можливий дохід	200	250	260	150	300
	Імовірність	0,3	0,3	0,2	0,1	0,1
4	Можливий дохід	120	100	200	150	130
	Імовірність	0,2	0,4	0,2	0,2	0,2
5	Можливий дохід	520	400	250	450	300
	Імовірність	0,6	0,1	0,1	0,1	0,1
6	Можливий дохід	520	400	200	450	300
	Імовірність	0,2	0,4	0,1	0,2	0,1
7	Можливий дохід	220	300	200	150	300
	Імовірність	0,2	0,3	0,2	0,2	0,1
8	Можливий дохід	100	300	200	150	300
	Імовірність	0,2	0,2	0,3	0,1	0,2
9	Можливий дохід	230	400	200	250	200
	Імовірність	0,1	0,3	0,2	0,2	0,2
10	Можливий дохід	220	120	200	350	400
	Імовірність	0,1	0,5	0,1	0,2	0,1

Яка можлива економічна інтерпретація цих зон?

ТЕМА 4
ЕКСПЕРТНІ СИСТЕМИ

1. Експертна система (ЕС) як програмний продукт.
2. Експертна система як система, що базується на знаннях.
3. Властивості ЕС.
4. Архітектура експертних систем.
5. Вимоги до ЕС.
6. Класифікація ЕС за призначенням.

1. Експертна система (ЕС) як програмний продукт

Найвідоміші практичні застосування штучного інтелекту – експертні системи. Дослідники у галузі ЕС часто використовують також термін «інженерія знань», введений Фейгенбаумом як «залучення принципів та інструментарію досліджень із ШІ для рішення прикладних проблем, що вимагають знань експертів».

Експертна система являє собою програмний продукт, що поводиться подібно до експерта в якійсь відносно вузькій предметній області. Їх створюють на основі глибоких спеціальних знань про певну предметну область (далі – ПрО), що отримують від експертів-спеціалістів. Це інтелектуальні програми, здатні робити логічні висновки на підставі знань у конкретній ПрО та забезпечувати рішення специфічних задач на професійному рівні. Основу ЕС складає база знань ПрО.

2. Експертна система як система, що базується на знаннях

Експертна система (далі – ЕС) – це складна інформаційна система, що оперує знаннями в певній ПрО з метою надання рекомендацій або вирішення проблеми.

ЕС акумулює знання експертів, щоб надати можливість використовувати їх менш кваліфікованим користувачам. Вона на основі обробки цих знань може давати інтелектуальні поради, приймати рішення на рівні експерта-професіонала, а також пояснювати процес знаходження того або іншого рішення.

Система, що базується на знаннях (knowledge-based system) – це система, процес роботи якої пов’язаний з використанням символічного подання знань та правил їх обробки, а не чітких алгоритмів. ЕС відносять до класу систем, що базуються на знаннях, але, крім того, мають давати конкретний результат за певний час та з потрібною достовірністю.

3. Властивості ЕС

Експертним системам властиві:

– спеціалізація у певній ПрО;

– використання БЗ;

– уміння приймати рішення за неповними або неточними даними;

– уміння пояснювати свої дії при розв’язуванні задачі та доводити їх обґрунтованість;

– здатність до розширення та нарощування функцій;

– здатність імітувати діяльність експерта;

– використання для розв’язування задач евристик – методів, що спираються на досвід та знання експерта, але, на відміну від алгоритмів, не завжди гарантують успіх.

ЕС мають широке практичне застосування. Розроблені ЕС у науці (класифікація тварин і рослин за видами), у медицині (постановка діагнозу, аналіз електрокардіограм, визначення методів лікування), у техніці (пошук несправностей у технічних пристроях, спостереження за польотом космічних кораблів і супутників), у соціології, криміналістиці, лінгвістиці тощо. Проте складність та висока вартість ЕС, а головне – їхня вузька спеціалізація – стримують їх поширення.

ЕС – це програма, що поводиться як експерт у певній вузькій ПрО. ЕС мають розв’язувати задачі з невизначеними і неповними вихідними даними, застосовуючи для цього експертні знання. Ці системи надають пояснення своєї поведінки і свого рішення, але відповідальність за прийняте рішення несе особисто користувач.

Оцінка ЕС користувачем значною мірою залежить від того, наскільки праця з експертною системою схожа на співробітництво

з експертом, і, відповідно, істотно залежить від якості пояснень, які система надає користувачеві у відповідь на його запитання. Важлива риса ЕС – їх адаптивність, тобто здатність до навчання.

4. Архітектура експертних систем

Прийнято виділяти в ЕС три основні модулі (рис. 7.7):

- база знань;
- машина виведення;
- інтерфейс із користувачем.

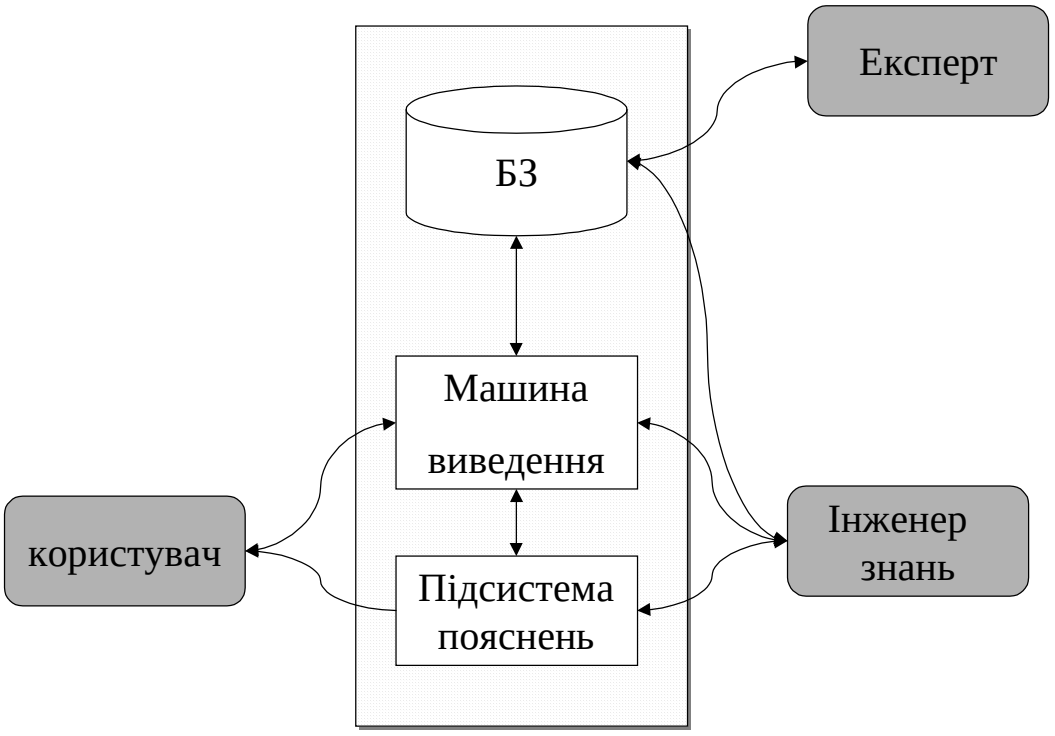

Рис. 4.1. Структура експертної системи

Користувач – спеціаліст у певній ПрО з відносно низькою кваліфікацією, для якого призначена ЕС. *Інженер знань* (далі – ІЗ) – спеціаліст у ШІ, що виступає посередником між експертом та БЗ, яку він створює.

Машина виведення – це програма, що моделює хід міркувань експерта.

Інтелектуальний редактор БЗ – це програмна компонента ЕС, яка надає ІЗ можливість створення БД у діалоговому режимі.

База знань ЕС – це сукупність відомостей про ПрО, для якої розробляється ЕС. Для функціонування ЕС її БЗ має бути наповнена знаннями.

Для цього запрошують *експертів* – висококваліфікованих спеціалістів у тій галузі, для якої розробляється ЕС. Їх завдання – формально описати всі свої знання, потрібні для функціонування ЕС. У БЗ містяться знання двох типів:

1) загальновідомі факти, явища, закономірності даної ПрО;

2) набір емпіричних правил, за якими спеціалісти приймають рішення за умов невизначеності, неповноти та суперечливості інформації.

У процесі розроблення такої БЗ можна виділити три основні фази: попередню, початкову і накопичувальну.

На *попередній* фазі ІЗ отримує від експерта або з інших джерел загальні відомості про ПрО (основні поняття, відношення, структуру даних) і формує загальне уявлення про принципи побудови ЕС, а потім обирає інструментарій для створення ЕС (приміром, порожню ЕС або мову подання знань) та середовища розроблення.

На *початковій* фазі ІЗ заповнює систему знаннями, що визначають організацію, структуру і спосіб подання БЗ.

Накопичувальна фаза характеризується набуттям основних знань про ПрО та передбачає виявлення неповноти, некоректності або суперечливості знань ЕС та здобуття знань, що усувають ці проблеми, а також надання цим знанням вигляду, зрозумілого ЕС.

Набуття знань передбачає спільну роботу ІЗ з експертами, які часто обґрунтовують свої висновки загальними концепціями, не виявляючи деталей, посилаються на інтуїцію та досвід, який базується на великій кількості взаємозалежних фактів, закономірностей і навичок.

Вирішити задачу створення БЗ ПрО можна двома шляхами, зробивши акцент або на процес *інтерн'ювання* експерта з наступним аналізом протоколу його міркувань, або на створення засобів *автоматизованого* здобуття знань та індуктивного генерування правил для конкретної ПрО.

Усі знання в БЗ поділяються на інтенсіональні та екстенсіональні. Інтенсіональні знання – це концептуальні знання про об’єкти ПрО та зв’язки між ними, а екстенсіональні (конкретні) знання – це кількісні характеристики інтенсіональної частини знань, тобто БД ЕС.

5. Вимоги до ЕС

Вимоги до ЕС:

компетентність – у конкретній ПрО ЕС має досягати того ж рівня, що і спеціалісти – люди;

ЕС має користуватися тими ж евристичними прийомами, так само глибоко і широко відображати *символьні міркування* (знання, на яких базується ЕС, подають у символьному вигляді поняття реального світу, міркування також відбуваються як перетворення символьних наборів);

глибина – експертиза має вирішувати серйозні, нетривіальні задачі, що відрізняються складністю знань, які використовує ЕС, або великим обсягом інформації; це не дозволяє використовувати повний перебір варіантів для розв’язання задачі і змушує вдаватися до евристичних методів;

самосвідомість – ЕС має містити в собі механізм пояснення того, яким чином вона приходить до розв’язання задачі.

6. Класифікація ЕС за призначенням

ЕС, що *виконують інтерпретацію*, як правило, використовують інформацію від датчиків для опису ситуації. Приміром, це може бути інтерпретація показників вимірювальних приладів на заводі для визначення стану процесу. Такі ЕС мають справу не з чіткими символьними поданнями проблемної ситуації, а безпосередньо з реальними даними. Це призводить до ускладнень, яких немає в інших ЕС, тому що їм доводиться обробляти інформацію недостатню, неповну, ненадійну або помилкову. Вони використовують спеціальні методи реєстрації характеристик безупинних потоків даних, сигналів або зображень і методи їх символьного подання.

ЕС, що *здійснюють прогноз*, визначають імовірні наслідки заданих ситуацій. Прикладами проблем, що вирішують такі ЕС,

служать прогноз збитків врожаю від деяких видів шкідливих комах, оцінка попиту на нафту на світовому ринку, прогнозування місця виникнення наступного збройного конфлікту на підставі даних розвідки. Системи прогнозування іноді використовують імітаційне моделювання, тобто програми, що відображають причинно-наслідкові взаємозв'язки в реальному світі, щоб генерувати ситуації або сценарії, які можуть виникнути за тих або інших вхідних даних. Можливі ситуації разом із знаннями про процеси, що породжують ці ситуації, утворюють передумови для прогнозу.

Досі розроблено порівняно мало систем для прогнозування, можливо, тому, що досить важко взаємодіяти з імітаційними моделями і створювати їх.

ЕС виконують *діагностику*, використовуючи описи ситуацій, характеристики поведінки або знання про конструкцію компонентів, щоб установити ймовірні причини неправильного функціонування системи. Прикладами служать визначення причин захворювання за симптомами, що спостерігаються в пацієнтів; локалізація несправностей в електронних схемах і визначення відмов у системі охолодження ядерних реакторів. Діагностичні системи часто є консультантами, що не тільки ставлять діагноз, але і допомагають у налагоджуванні. Вони можуть взаємодіяти з користувачем, щоб надати допомогу в пошуку несправностей, а потім запропонувати порядок дій з їх усунення. Медицина – це цілком природна галузь для діагностики, і дійсно, у ній було розроблено більше діагностичних систем, чим у будь-якій іншій ПрО, проте зараз багато діагностичних систем розробляють для застосування в інженерній справі та комп'ютерних системах.

ЕС, що *виконують проектування*, розробляють конфігурацію об'єктів, враховуючи обмеження ПрО. Прикладами можуть служити генна інженерія та синтез складних органічних молекул.

ЕС, зайняті *плануванням*, проектують дії; вони визначають повну послідовність дій перед тим, як починається їх виконання. Прикладами можуть служити створення плану використання послідовності хімічних реакцій для синтезу складних органічних сполук або створення плану повітряного бою з метою нейтралізації певного чинника боєздатності ворога.

ЕС, що *виконують спостереження*, порівнюють справжню поведінку системи з її очікуваною поведінкою. Приміром, спостереження за показаннями вимірювальних приладів у ядерних реакторах мають виявляти аварійні ситуації, а оцінка даних моніторингу хворих у блоках інтенсивної терапії – небезпеку для життя людини. ЕС порівнюють результати спостереження з даними, що притаманні стандартним ситуаціям. Такі ЕС за самою своєю природою мають працювати в режимі реального часу і здійснювати залежну як від часу, так і від контексту інтерпретацію поведінки об’єкта спостереження.

ЕС, що *навчають*, аналізують та коригують поведінку того, кого навчають. Ці системи створюють модель знань того, хто навчається, і модель того, як він ці знання застосовує до рішення проблеми. ЕС діагностує помилки і вказує на них, здійснює аналіз і будує плани виправлень цих помилок.

ЕС, що *здійснюють управління*, адаптивно управляють поведінкою системи в цілому. Приклади – управління виробництвом і розподілом комп’ютерних систем або контроль за станом хворих при інтенсивній терапії. Такі ЕС мають містити компоненти спостереження, щоб відслідковувати поведінку об’єкта.

«*Порожні*» ЕС – це інструментальні засоби для побудови інших ЕС. Вони не містять конкретних правил ПрО. Прикладом такої системи є інструментальний комплекс ІндЕкс, розроблений автором даної глави, що призначається для автоматичної розробки консультуючих систем. Цей комплекс містить бібліотеку алгоритмів індуктивного здобуття знань, підсистему візуалізації дерева рішень у вигляді Р-графів, інтелектуальний користувацький інтерфейс, засоби пояснення та механізм інтерпретації дерева рішень. Система здатна обробляти неповні та нечіткі дані. За допомогою ІндЕкс були створені прикладні ЕС для оцінки якості геологічних досліджень, для прогнозування масового розмноження шкідливих комах, для оптимізації медико-біологічних досліджень осіб, що потерпіли внаслідок аварії на Чорнобильській АЕС, економічного прогнозування тощо.

Інструментальні засоби створення ЕС:

– символьні мови програмування, орієнтовані на створення ЕС і систем штучного інтелекту;

– мови інженерії знань, тобто мови високого рівня, орієнтовані на побудову ЕС (приміром, OPS-5, Пролог);

– системи, що автоматизують розробку систем ШІ, орієнтованих на знання;

– оболонки ЕС.

За зв’язком з реальним часом ЕС поділяють на:

– статичні;

– квазідинамічні;

– динамічні.

Статичні ЕС розробляються для ПрО, в яких БЗ та дані, що вона інтерпретує, стабільні та незмінні. Приклад статичної системи – ЕС для діагностики несправностей автомобіля.

Квазідинамічні ЕС здатні інтерпретувати ситуацію, що змінюється за певний інтервал часу. Приклад квазідинамічної системи – ЕС для обробки лабораторних вимірів технологічного процесу.

Динамічні ЕС здатні обробляти інформацію від датчиків у режимі реального часу. Приміром, динамічні ЕС використовують для управління гнучкими виробничими комплексами.

Отже, аналіз галузей використання ЕС дозволяє визначити тенденції їх розвитку для застосування в динамічному гетерогенному розподіленому середовищі. Це потребує створення та застосування нових інформаційних технологій, що базуються на знаннях, та Web-середовища. Найбільший інтерес з цієї точки зору на сьогодні представляють агентні технології та семантичні Web-сервіси, які прискорюють та полегшують розробку інтелектуальних програмних продуктів.

Питання для самоконтролю

1. Що таке база знань і як вона використовується в ЕС?
2. Опишіть ЕС як програмний продукт.
3. Експертна система як система, що базується на знаннях.
4. Які властивості ЕС?
5. Яка архітектура експертних систем?
6. Які вимоги до ЕС?
7. Виконайте класифікацію ЕС за призначенням.

ТЕМА 5
ОНТОЛОГІЧНИЙ АНАЛІЗ ТА УПРАВЛІННЯ ЗНАННЯМИ В WEB

1. Онтології як засіб представлення знань у Web.
2. Формальна модель онтології і приклади її застосування.
3. Основні характеристики онтології.
4. Дескриптивна логіка як теоретичний базис для онтологічного представлення знань.
5. Інструментальні засоби побудови онтологій.
6. Методології розробки онтологій.
7. Методології розвитку онтологій.
8. Загальні етапи побудови онтології.

1. Онтології як засіб представлення знань у Web

Філософський термін «*онтології*» зараз активно використовують в ІТ. Множина об’єктів і відношення між ними відображаються у словнику, в якому система, заснована на знаннях, подає ці знання.

У найбільш загальному випадку онтологія – це угода про спільне використання понять, що містить засоби подання предметних знань і домовленості про методи міркувань. Неформально онтологія являє собою певний опис погляду на світ у конкретній сфері інтересів. Цей опис складається з набору термінів і правил використання цих термінів, що обмежують їх значення в рамках конкретної ПрО.

На формальному рівні онтологія – це система, що складається з наборів понять і тверджень про ці поняття, на основі яких можна будувати класи, об’єкти, відношення, функції та теорії. Онтологія, як зразок загальної угоди про семантику ПрО, сприяє встановленню коректних зв’язків між значеннями елементів ПрО, створюючи умови для їх спільного використання. Онтології можна застосовувати як будівельні блоки компонентів БЗ, словник для зв’язку між агентами, визначення класів для програмних систем тощо.

Онтології – це бази знань спеціального виду, які містять семантичну інформацію з певної ПрО. Основні завдання, що можуть успішно вирішуватися за допомогою онтологій:

– подання знань для виведення інформації, релевантної запиту користувача;

– фільтрація та класифікація інформації;

– індексування зібраної інформації;

– організація спільної термінології для комунікації між користувачами.

Онтології використовуються в електронній комерції для підтримки автоматизованого обміну даними між покупцями і продавцями, для вертикальної інтеграції ринків, а також для повторного використання описів різними електронними торговельними точками. Механізми пошуку також застосовують онтології для знаходження сторінок із синтаксично різними, але семантично однаковими словами.

Наведені нижче визначення та приклади використання онтологій базуються на матеріалах, що містяться в [4]. Онтологія (від др.-грец. «*онтос*» – «буття», «*логос*» — «вчення», «поняття») – термін, що визначає вчення про буття, на відміну від *гносеології* – вчення про пізнання. Предметом онтології як частини метафізики було вивчення абстрактних і загальних філософських категорій, таких як *буття, субстанція*, *причина, дія, явище* тощо, а сама онтологія як наука претендувала на повне пояснення причин усіх явищ.

Для практичного використання краще застосовувати визначення онтології, запропоноване FIPA (Foundation for Intelligent Physical Agents): *онтологія* – це система категорій, що є наслідком певного погляду на навколишнє середовище.

При цьому сама система категорій не залежить від конкретної мови: онтологія завжди та ж сама незалежно від мови, що використовується для її опису.

У ШІ звичайно використовують таке визначення:

Онтологія – це знання, формально подані на основі концептуалізації, що припускає опис множини об’єктів і понять, знань про них та зв’язків між ними.

Формально онтологія складається з термінів, організованих у таксономію, їх визначень і атрибутів, а також пов’язаних з ними аксіом і правил виведення.

Часто набір припущень, що складають онтологію, має форму логічної теорії першого порядку, де терміни словника є іменами унарних та бінарних предикатів, які називають відповідно концептами і відношеннями. У найпростішому випадку онтологія описує тільки ієрархію концептів, пов’язаних відношеннями категоризації. У більш складних випадках до неї додаються певні аксіоми для відображення інших відношень між концептами і для того, щоб обмежити їх інтерпретацію. Онтологія являє собою БЗ, що описує факти, які вважаються завжди істинними в рамках визначеного угруповання на основі загальновживаного значення словника, що використовується.

Можна визначити онтологію як експліцитну специфікацію певної теми. Такий підхід припускає формальне і декларативне подання певної теми, яке містить словник (або список констант) для відсилання до термінів ПрО, обмеження цілісності на терміни, логічні твердження, що обмежують інтерпретацію термінів і те, як вони зіставляються один з одним.

2. Формальна модель онтології та приклади її застосування

Розглянемо формальну модель онтології та приклади її застосування:

Формальна модель онтології О – це упорядкована трійка

$$\text{О} = \langle \text{Х}, \Re, \Ф \rangle,$$

де Х – скінченна множина термінів (понять) ПрО, яку описує онтологія О;

$\Re$ – скінченна множина відношень між термінами заданої ПрО;

$\Ф$ – скінченна множина функцій інтерпретації, заданих на термінах і/або відношеннях онтології О.

Якщо $\Phi = \varnothing$ і $\Re = \varnothing$, тоді онтологія О трансформується в простий словник: $O = V = \langle X, \{\ \}, \{\ \} \rangle$. Така вироджена онтологія може бути корисна для підтримки словників ПрО, але через те, що вона не дозволяє експліцитно вводити значення термінів, її застосовують, якщо терміни належать дуже вузькому (приміром, технічному) словнику і їхні значення вже заздалегідь добре узгоджені в межах певного (приміром, наукового) співтовариства. Відомими прикладами онтології цього типу є індекси пошукових машин в Інтернет.

Якщо $\Re = \varnothing$, але $\Phi \neq \varnothing$, тоді кожному елементу множин термінів із X може бути поставлена у відповідність функція інтерпретації f із Φ.

Спеціальний підклас онтології – проста *таксономія*: $O = T^0 = \langle X, \{"\text{бути\_елементом\_класу}"\}, \{\ \} \rangle$. Це ієрархічна система понять, пов'язаних між собою відношенням «бути елементом класу», яке має фіксовану семантику і дозволяє організовувати структуру понять онтології у вигляді дерева. Такий підхід має свої переваги і недоліки, але в загальному випадку є адекватним і зручним для представлення ієрархії понять.

Залежно від призначення розрізняють такі онтології:

– онтології верхнього рівня, що містять найбільш загальні терміни;

– онтології, орієнтовані на конкретну задачу;

– прикладні онтології.

3. Основні характеристики онтології

Основні характеристики онтології зображено на рис. 5.1.

Чіткість – онтологія має ефективно відображати зміст термінів. Визначення мають бути об'єктивними, хоч мотивація використання термінів може визначатися ситуацією або вимогами обчислювальної ефективності. Для об'єктивізації визначень використовують чітко фіксований формалізм, при цьому доцільно задавати визначення у вигляді логічних аксіом.

Узгодженість – усі визначення мають бути логічно не суперечливими, а усі твердження, що виводяться в онтології, мають не суперечити аксіомам.

Розширюваність – онтологія має бути спроектована так, щоб забезпечувати використання словників термінів, що припускають можливість монотонного розширення та спеціалізації без необхідності ревізії вже існуючих понять.

Мінімум впливу кодування – концептуалізація, що лежить в основі створюваної онтології, має бути специфікована на рівні подання, а не символьного кодування.

Мінімум онтологічних зобов'язань – онтологія має містити тільки найбільш істотні припущення про ПрО, що вона моделює, щоб залишати простір для розширення і спеціалізації.

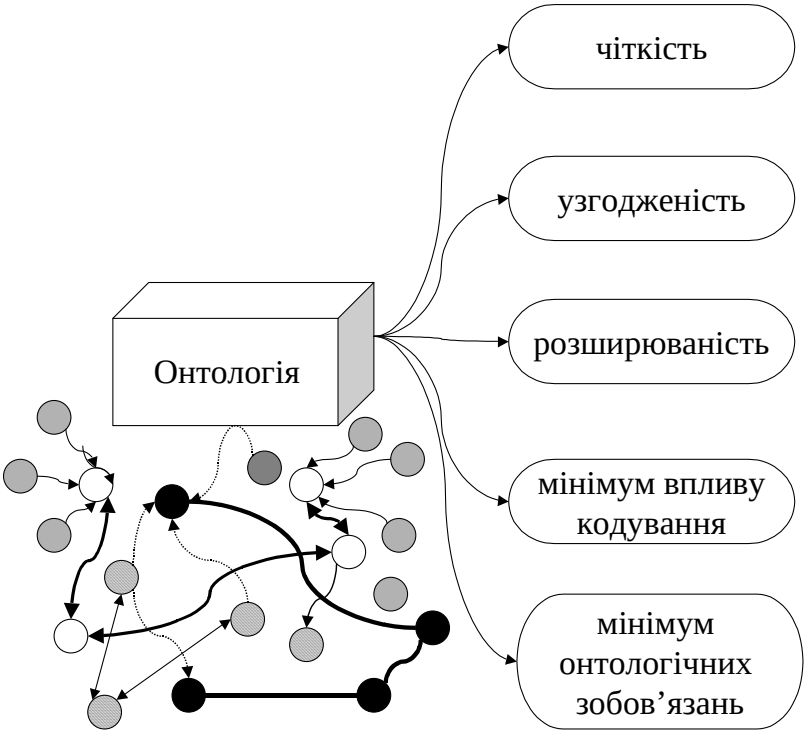

Рис. 5.1. Основні властивості онтології

Методологія побудови онтологій потребує, у першу чергу, позначення цілей і області застосування онтології. У процесі побудови онтології фіксують знання про ПрО, а саме:

– визначають основні поняття та їхні відношення в обраній ПрО;

– створюють точні несуперечливі визначення для кожного основного поняття і відношення;

– визначають терміни, що пов’язані з цими термінами і відношеннями.

На етапі кодування сукупності основних термінів, що використовують в онтології, поділяють на окремі класи понять; обирають мову подання онтології та безпосередньо задають фіксовану концептуалізацію обраною мовою подання знань.

Процес побудови онтології, відповідно до методології стандарту онтологічних досліджень IDEF5 (INTEGRATED DEFintion), складається з п’яти основних дій:

1) вивчення і систематизування початкових умов для встановлення основних цілей і контексту проекту розробки онтології;

2) збір та накопичення необхідних початкових даних для побудови онтології;

3) аналіз та групування зібраних даних для полегшення побудови термінології;

4) початковий розвиток онтології;

5) уточнення та затвердження онтології – заключна стадія процесу.

Нині існує багато проектів, в основу яких покладені онтології: TOVE (Toronto Virtual Enterprise), Ontolingua, KACTUS (в складі проекту ESPIRIT), SHOE (Simple HTML Ontology Extensions), Plinus, Cyc (MCC), $(KA)^2$, Semantic Web (консорціум W3C) тощо.

Для рішення задачі семантичної інтероперабельності Web у складі проекту Semantic Web розробляється мова подання онтологій OWL (OWL Web Ontology Language), що походить від DAML+OIL Web Ontology Language та базується на RDF (Resource Description Framework). База знань OWL – це набір трійок, яким призначаються певні значення. Онтологія OWL містить кілька компонентів: заголовків (headers), елементів класів (class elements), елементів властивостей (property elements) та прикладів (instances), що можуть повторюватися. Онтологія – це послідовність аксіом та фактів, яка може також містити посилання на інші онтології. Крім того, до онтології може входити допоміжна інформація – приміром, про її розробника.

Мета міжнародного проекту $(KA)^2$ – інтелектуальний пошук в Інтернет та автоматичне здобуття нових знань. У рамках ініціативи $(KA)^2$ виділяються три основних напрями досліджень:

• онтологічний інжиніринг (ontological engineering);

• анотація Web-сторінок;

• запити до Web-сторінок і виведення відповідей на базі онтологічних знань.

Для специфікації онтології розроблена спеціальна мова подання знань. Підмножина цієї мови служить і для формулювання запитів, а мова анотування – для супроводження Web-документів онтологічною інформацією. Для правильного формулювання запиту необхідно знати, які концепти присутні в онтології і які атрибути мають концепти. Тому система подання онтологічних знань надає своїм користувачам засоби візуалізації онтології.

Проект SHOE орієнтований на рішення проблеми додавання до Web-сторінок семантичної інформації і співвіднесення її з онтологіями відповідних ПрО. Передбачається, що, використовуючи цю інформацію, пошукові системи зможуть забезпечувати більш релевантні відповіді на запити. Дослідження у проекті SHOE ведуться у двох напрямах:

– розробка множин повторно використовуваних онтологій (reusable ontologies) для концептів, що часто зустрічаються у Web-ресурсах;

– створення засобів проектування онтології – анотаторів знань (Knowledge Annotator).

4. Дескриптивна логіка як теоретичний базис для онтологічного представлення знань

Дескриптивні логіки (далі – DL) виникли як розширення фреймів та семантичних мереж механізмами формальної логіки. Зараз DL використовуються в Semantic Web для побудови онтологій. DL – це сімейство мов представлення знань, що дозволяють описувати поняття предметної області у формалізованому вигляді. Будь-яка логіка DL є логікою першого порядку, але не навпаки. Базові елементи DL – це множина класів NC; Множина індивідуумів NI; Множина відношень NR.

Дескриптивні логіки дозволяють описувати поняття ПрО в недвозначному, формалізованому вигляді. Вони поєднують у собі, з одного боку, досить багаті виразні можливості, а з іншого – задовільні обчислювальні властивості, такі як можливість вирішення і відносно невисока обчислювальна складність основних логічних проблем, що уможливлює їхнє застосування на практиці. Таким чином, DL являють собою компроміс між виразністю і можливістю вирішення. DL можна розглядати як вирішувані фрагменти логіки предикатів, синтаксично ж вони близькі до модальних логік.

У математичній логіці всяка мова характеризується своїм *синтаксисом*, тобто правилами побудови виразів цієї мови, і *семантикою*, тобто способом приписування цим виразам деякого формального значення, наприклад, указівкою, які вирази вважаються правдивими і помилковими. Щоб сформулювати синтаксис якої-небудь DL, необхідно задати непорожні (і звичайно скінчені) множини символів – так званих атомарних концептів і атомарних ролей – з яких будуватимуться вирази мови даної логіки. Конкретна DL характеризується набором конструкторів та індуктивним правилом, за допомогою якого складені концепти даної логіки будуються з атомарних концептів і атомарних ролей, використовуючи ці конструктори.

Мови та стандарти подання онтологій

Мови подання онтологій, які існують на цей момент, можуть бути класифіковані як мови на основі XML або не-XML-орієнтовані.

XML-орієнтовані мови отримали більш широкий розвиток для Semantic Web, оскільки XML найбільш добре підходить для рішення питання інтероперабельності, що є одним з основних вимог Semantic Web. У Таблиці 1 надані стислі описи деяких з XML-орієнтованих мов подання онтологій.

RDF(S) є найбільш простою та найбільш маштабованою XML-мовою подання онтологій. Вона базується на обчисленні предикатів і використовує для визначення семантики трійки «суб’єкт, предикат і об’єкт». Однак вона не достатньо виразна і може використовуватися тільки для вказівки концептів та бінарних відношень між ними.

Мова OWL розширює RDF(S) і забезпечує конструкції для вираження понять, відношень, потужності, анотацій та конкретизації понять та ін. Крім того, OWL підтримується великою кількістю редакторів онтологій і вирішувачей. Усе це дозволяє називати OWL найбільш підходящою мовою для Semantic Web.

5. Інструментальні засоби побудови онтологій

Серед інструментальних засобів, що використовуються в онтологічному аналізі, можна виділити засоби, орієнтовані на підтримки різних етапів життєвого циклу онтології та на різні рівні обробки онтологічних структур. По-перше, це редактори онтологій, призначені для створення і модифікації онтологій користувачем. По-друге, це програмні засоби, призначені для виконання логічних запитів до онтологій, різноманітних операцій над онтологіями (приміром, перевірки їх на непротирічність або їх інтеграції). По-третє, це засоби, орієнтовані на візуалізацію онтологічних знань та представлення їх у зручній для користувача формі. У різних програмних засобах такі функції можуть поєднуватися у різних комбінаціях. Зараз розроблено велику кількість інструментів для роботи з RDF і OWL. До них відносять редактори, CMS, середовища розробки, репозиторії RDF, генератори RDF, валідатори, сервери SPARQL і механізми логічного виведення.

Системний аналіз онтологічних сервісів, необхідних для ефективного керування знаннями в Semantic Web та розгляд послуг, як підтримують розповсюджені сьогодні програмні засоби для роботи з онтологіями (редактори, ризонери, візуалізатори тощо), показують, що всі онтологічні сервіси можна поділити на кілька груп відповідно до того, які функціональні можливості потрібні користувачам:

– сервіси представлення та візуалізації онтологій, поданих різними засобами та мовами, – потрібні у всіх системах, які використовують онтологічне подання знань;

– сервіси редагування онтологій, які забезпечують створення нових онтологій, редагування наявних онтологій, операції над їх класами, властивостями класів та екземплярами класів, у тому

числі й такі, які забезпечують сумісну роботу над онтологіями (потрібні в тих системах, які не тільки використовують готові онтології, але й створюють свої онтології або модифікують наявні);

– сервіси логічного виведення над онтологіями, аналогічні тим функціональним можливостям, що надають різонери, у тому числі – сервіси перевірки онтології на узгодженість;

– сервіси автоматизованого поповнення онтологій (приміром, шляхом обробки природномовних текстів, індуктивного та традуктивного узагальнення відомостей з баз даних, обробки метаданих) – у тому випадку, якщо подібні сервіси потрібні у відповідній предметній області та реалізовані програмно;

– сервіси операцій над онтологіями, що забезпечують їх порівняння та інтеграцію, – якщо для предметної області створено та програмно реалізовано відповідні алгоритми; сервіси підтримки виконання запитів SPARQL.

Проведені вивчення загальних програмних застосувань дозволяють зробити висновок про те, що, незважаючи на те, що кількість засобів Semantic Web стрімко зростає, багато областей семантичних задач досі ще не охоплено. Наприклад, необхідно автоматизувати порівняння онтологій, щоб визначити однакові для обох онтологій класи і властивості; перетворення представлення онтології з реляційної БД в OWL; розподілене збереження онтологічних даних з можливістю виконання запитів та візуалізацію онтологічних даних.

Редактор онтологій – це програмний засіб, який підтримує створення онтологій та їхнє редагування. Така програма надає можливості роботи з однією чи кількома мовами представлення онтологій, імпорт/експорт у різні формати, доступ до бібліотек онтологій, візуалізацію, засоби логічного висновку, мови запитів, зв’язок з іншими онтологіями тощо. Сьогодні розроблено багато програмних засобів, призначених для обробки онтологічних знань, що робить актуальною задачу вибору редактора онтологій, що найбільш відповідає специфіці задачі, для якої розробляється онтологія, і визначення критеріїв, що впливають на цей вибір.

Стенфордському університету належить розробка трьох широко відомих редакторів онтологій – *Protégé*, *Chimaera* та *Ontolingua*. Найбільше поширення з них отримав Protege.

Більшість редакторів підтримує кілька форматів опису онтологій, що обумовлено відсутністю єдиного широко використовуваного формату і залишає право вибору за користувачем. Останнім часом стає обов'язковим для редакторів підтримувати виконання логічного виведення, а саме – перевірки несуперечності побудованої моделі і забезпечення можливості виконання запитів. Позитивним фактором є використання в редакторах онтологій реляційних баз даних, що дозволяє зробити роботу редактора більш стійкою, збільшити швидкість роботи редактора, значно підвищити максимальну кількість об'єктів, з якими здатний працювати редактор, і зменшити час відновлення. Можна сформулювати три критерії вибору редактора онтологій:

Якщо визначена базова онтологічна модель, то варто використовувати спеціалізовані критерії.

Якщо необхідно забезпечити підтримку і можливість виконання імпорту/ експорту онтологій у різні формати, то як альтернативи варто розглядати універсальні редактори. Для спеціалізованих редакторів характерна підтримка обмеженої кількості форматів, як правило, формату розширюваної мови розмітки (XML) і формату мови онтології для Інтернет (OWL-RDF).

Якщо необхідно забезпечити можливість виконання запитів до онтологічної бази знань, то варто використовувати спеціалізований редактор. Через те, що спеціалізовані редактори розробляються на основі базової онтологічної моделі, існує можливість забезпечити ефективний механізм виконання запитів.

Сьогодні при створенні онтологій найбільше часто застосовується такий програмний засіб, як Protégé, що є безкоштовним і має багато корисних властивостей. У зв'язку з цим розглянемо детальніше можливості, які надає Protégé.

Protégé – це вільний, відкритий редактор онтологій, призначений для побудови баз знань. Платформа Protégé підтримує два основні способи моделювання онтологій: за допомогою редакторів Protégé-Frames і Protégé-OWL. Онтології, побудовані в

Protégé, можуть бути експортовані в багато різних форматів, включаючи RDF та OWL.

Protégé має відкриту, легко розширювану архітектуру за рахунок підтримки модулів розширення функціональності. Protege підтримується великим науковим співтовариством, що складається з розроблювачів і учених, урядових і корпоративних користувачів, що використовують його для рішення задач, пов'язаних зі знаннями, у таких різноманітних областях, як біомедицина, обробка знань і корпоративне моделювання.

Модель Protégé Frames була спеціально розроблена для використання в редакторі онтологій Protégé. Вона базується на фреймовому представленні і протоколі OKBC (Open Knowledge Base Connectivity). Відомі й інші застосовувані на практиці моделі, що активно використовуються в онтологічному інжинірингу, особливе місце серед яких займає мова опису онтологічних знань OWL, що прийнята як стандарт консорціумом W3C. Однак усі сучасні практично-орієнтовані моделі еквівалентні формальній моделі <X,RF) або є її обмеженими представленнями, а отже, можуть бути представлені в ній, наприклад, відповідно до схеми співвіднесення загальноприйнятої моделі і фреймової моделі Protégé. Виходячи з цього, для опису операцій над онтологіями можна використовувати кожну з моделей. Фреймова модель Protege має високий ступінь абстракції, досить поширена й описана в багатьох публікаціях і документації розроблювачів, має формальні засоби опису в мові OWL, і, крім того, редактор, заснований на цій моделі, є загальнодоступним як у вихідних кодах, так і в бінарному вигляді для більшості сучасних платформ, що дає можливість застосовувати його на практиці.

Сьогодні редактор онтологій Protégé можна за кількома розглянутими вище критеріями визнати найкращим програмним засобом, призначеним для цих цілей. Крім того, плагіни Protégé виконують багато функцій, характерних для інших типів інструментальних засобів, пов'язаних з обробкою онтологій, приміром, ризонери HermiT, Snorocket, візуалізатори, анотатори, конвертори форматів та мов подання.

6. Методології розробки онтологій

Онтологічний аналіз звичайно починається зі складання словника термінів, які використовуються під час обговорення, та дослідження характеристик об’єктів і процесів, що становлять розглянуту систему, а також створення системи точних визначень цих термінів. Крім того, документуються основні логічні взаємозв’язки між відповідними введеним термінам поняттями. Надалі не робитимемо розходження між поняттями й термінами. Результатом цього аналізу є онтологія системи або сукупність словника термінів, точних їх визначень і взаємозв’язків між ними.

Таким чином, онтологія містить у собі сукупність термінів і правила, згідно з яким ці терміни можуть бути скомбіновані для побудови достовірних тверджень про стан розглянутої системи в деякий момент часу. Крім того, на основі цих тверджень можуть бути зроблені відповідні висновки, які дозволять вносити зміни в систему для підвищення ефективності її функціонування.

При побудові онтології в першу чергу відбувається створення списку або бази даних дескрипторів і за допомогою їх, якщо їхній набір достатній, створюється модель системи. Таким чином, на початковому етапі повинні бути виконані такі задачі:

1) створення й документування словника термінів;

2) опис правил і обмежень, згідно з яким на базі введеної термінології формуються достовірні твердження, які описують стан системи;

3) побудова моделі, яка на основі існуючих тверджень дозволяє формувати необхідні застосовувані твердження.

7. Методології розвитку онтологій

Розробка онтологій відома під назвами інженерії онтологій (ontology engineering) або побудова онтологій (ontology building), а також можуть бути досліджені під рубрикою – онтологічне навчання (ontology learning).

У простішому випадку побудова онтології зводиться до:

1) виділення базових концептів ПрО;

2) побудови зв’язків між концептами – визначення відношень і взаємодій між базовими концептами.

Процес побудови онтологій може бути висхідним або спадним. Однак у зв’язку з тим, що висхідний підхід надзвичайно трудомісткий і доки не існує засобів, які дозволили б створити повну систему знань («модель миру»), в основному застосовується спадний підхід, орієнтований на конкретні, часто дуже обмежені практичні задачі. Таким чином, існує проблема створення онтологій у вузькій предметній області, що порушує питання про створення чотирьох останніх рівнів ієрархії. Загалом застосовуються такі підходи до побудови онтологій: Cyc method, Methology by Uschold і King, METHONTOLOGY, On-To-Knowledge тощо. Вони описують процеси побудови онтології, пропонують спосіб виділення понять предметної області, певну модель життєвого циклу розробки онтології, спосіб формалізації знань.

Проведений аналіз стану розвитку методологічних досліджень показує, що темпи розвитку нових методологій за останні п’ять років зменшилися, хоча проблема створення онтологій до сих пір залишається відкритою, а загальноприйнятої методології, яка б підходила для більшості задач, поки немає. Крім того, існуючі методології орієнтовані на спеціалістів з онтологічного аналізу, а не на спеціалістів прикладних областей, і потребують значних спеціальних навичок для їх ефективного використання.

Існуючі методології розробки онтологій можна поділити на прості, які надають тільки базові рекомендації для розробки онтологій (Sensus, Kaktus, Uschold and King, Gruninger and Fox, Noy and McGuinness тощо), і зрілі, що забезпечують деталізовані підходи до розробки онтологій на всіх етапах їх життєвого циклу (Methontology, OTK, UPON тощо). Крім того, іноді виділяють методології, які забезпечують розподілену розробку (приміром, HCOME і DILIGENT), але з нашої точки зору таке виділення не суттєве.

Методологія IDEF5

Методологія структурного аналізу і проектування SADT (Structured Analysis and Design Technique) породила цілий ряд методів моделювання IDEFx для задач інжинірингу та реінжинірин-

гу бізнес-процесів, у тому числі – IDEF5 (метод фіксації онтології), що забезпечує наочне подання даних, отриманих у результаті обробки онтологічних запитів у простій природній графічній формі. Методологія SADT використовує структурний підхід, тобто роздільну побудову моделі функцій (діаграми потоків даних) і моделі даних (діаграми «сутність – зв'язок»). Він складається з п'яти основних етапів:

1) вивчення й систематизування початкових умов. Ця дія встановлює основні цілі й контексти проекту розробки онтології, а також розподіляє ролі між членами проекту;

2) збору й накопичення даних. На цьому етапі відбувається збір і накопичення необхідних початкових даних для побудови онтології;

3) аналізу даних. Ця стадія полягає в аналізі і групуванні зібраних даних і призначена для полегшення побудови термінології;

4) початкового розвитку онтології. На цьому етапі формується попередня онтологія на основі відібраних даних;

5) уточнення й затвердження онтології. Заключна стадія процесу.

Процес побудови онтології, відповідно до методології IDEF5, полегшує збір знань про фізичні й концептуальні об'єкти разом з їхніми асоціаціями і забезпечує засоби для діаграмного відображення онтології.

Методологія METHONTOLOGY

Підхід до побудови й супроводу онтологій METHONTOLOGY базується на принципах Грубера і виділяє в «життєвому циклі» створення онтології керування проектом, розробку онтології та підтримку розробки.

Процедури керування проектом включають планування, контроль і гарантії якості. Планування визначає, які задачі повинні бути виконані, як вони організуються, як багато часу і яких ресурсів потрібно для їхнього виконання. Контроль гарантує, що заплановані задачі виконані й саме так, як це передбачалося. Гарантії якості потрібні для того, щоб бути впевненим у тім, що компоненти та продукт у цілому перебувають на заданому рівні.

Власне розробка включає специфікацію, концептуалізацію, формалізацію й реалізацію.

Відповідно до методології METHONTOLOGY спочатку необхідно побудувати глосарій термінів ПрО, а коли глосарій термінів досягає «істотного» об'єму, – дерева класифікації концептів для ідентифікації таксономій ПрО. Потім будують діаграми бінарних відношень для фіксації відношень між концептами однієї або різних онтологій, які можуть послужити базовим матеріалом для інтеграції різних онтологій. Для кожного дерева класифікації концептів будуються:

1. Словник концептів, який містить усі концепти ПрО, екземпляри таких концептів, атрибути екземплярів концептів, відношення, джерелом яких є концепт, а також (опціонально) синоніми й акроніми концепту.

2. Таблиця бінарних відношень для відношень, початкові концепти яких містяться в класифікаційному дереві, де фіксується його ім'я, імена концепту-джерела й цільового концепту, інверсне відношення тощо.

3. Таблиця атрибутів екземпляра для кожного екземпляру зі словника концептів (ім'я атрибута, тип значення, одиниця виміру, точність, діапазон зміни, значення «за замовчуванням», атрибути, які можуть бути виведені з використанням даного, формула виведення атрибута).

4. Таблиця атрибутів класу для кожного класу зі словника концептів з аналогічними характеристиками.

5. Таблиця логічних аксіом, у якій даються визначення концептів через завжди правдиві (true) логічні вирази. Визначення кожної аксіоми включає її ім'я, природно-язиковий опис, концепт, до якого аксіома відноситься, атрибути, використовувані в аксіомі, логічний вираз, який формально описує аксіому.

6. Таблиця констант, де для кожної константи вказується її ім'я, природно-язиковий опис, тип значення, саме значення, одиниця виміру, атрибути, які можуть бути виведені з використанням цієї константи, та ін.

7. Таблиця формул для кожної формули, яка включена в таблицю атрибутів екземпляра.

8. Дерева класифікації атрибутів, які графічно показують відповідні атрибути й константи, які використані для висновку значення кореневого атрибута й формули, які застосовані для цього. Загалом, ці дерева використовуються для перевірки того, що всі атрибути, які представлені у формулі, мають опис і жоден з атрибутів не пропущений.

9. Таблиця екземплярів для кожного входу в словник концептів.

Аналіз базової методології розвитку онтологій

Відповідно до аналізу існуючих онтологічних методологій розробка онтології складається з таких фаз:

Постановка задачі (предпроектний аналіз). Цей етап складається з виконання таких робіт. Необхідно визначити типи необхідних для виконання задачі онтологій, визначити інформаційні ресурси, які використовуватимуться у проекті, виявити необхідні вимоги до компетенцій спеціалістів, визначити критичні точки проекту і контрольні точки.

Специфікація технічних вимог (ТЗ). Аналіз і уточнення вимог, аналіз інформаційних ресурсів, складання специфікації вимог для різноманітних ролей, специфікація вимог до інструментів.

Вибір інформаційних ресурсів, пошук застосовуваних IP, дослідження і оцінка IP, виборка і кастомізація IP під вимоги. Необхідні інструментальні засоби: пошукові системи, обробники природної мови, системи розпізнавання текстів, системи обробки природної мови.

Вибір інструментальних засобів для навчання онтології: пошук інструментарію для навчання онтології, оцінка і вибір. Необхідні інструментальні засоби: інструментальні засоби навчання онтологій (TextToOnto, Text2Onto, OntoLT, OntoLearn та ін.).

Підготовка до навчання онтології. Призначення інструментів суб-доменам. Призначення (прив’язка) документів до інструментів. Специфікація виходів и точок взаємодії з користувачем. Специфікація методів для кожного інструментарію. Кастомізація обраних IP. Специфікація порядку виконання інструментів.

Створення адекватного середовища навчання онтологій. Виконання процесу навчання онтології за допомогою обраних інструментів. Виконання відповідних ручних коригувань (ручного вводу), необхідного для навчання онтологій при роботі інструментів. Оцінка і модифікація проміжних результатів. Реконфігурація набору інструментів. Необхідні інструментальні засоби: інструментальні засоби навчання онтологій (TextToOnto, Text2Onto, OntoLT, OntoLearn та ін.).

Оцінка й обчислення онтології

Інтеграція онтологій. Перевод у різноманітні формати подання. Інтегрування, об'єднання – злиття онтологій. Оцінка результатів. Результатом цього етапу буде кінцева онтологія. Необхідні інструментальні засоби: транслятори, інструменти злиття онтологій (PROMPT, AnchorPROMPT, GLUE, FCAMerge та ін.).

З нашої точки зору в цій методології не вистачає ще одного, останнього етапу, пов'язаного з розгортанням, тестуванням, розвитком онтологій та підтримкою версіонності і походження онтологій.

Необхідно констатувати, що всі існуючі на цей час методології побудови онтологій орієнтовані лише на досить вузьку конкретну задачу; застосовують свої, часто застарілі і закриті стандарти і підходи; а стандарти Semantic Web і мова OWL 2.0 не використовують; орієнтовані на спеціалістів-онтологів, а не експертів ПрО; носять занадто загальний характер і орієнтовані лише на розробку онтологій «верхнього рівня».

Таким чином, відкритою залишається потреба в простій узагальненій методології, придатної для використання неспеціалістами, яка б підтримувала сучасні стандарти керування знаннями і теорію відкритого світу і базувалася на новітніх технологіях програмування (відкриті та гетерогенні джерела даних, Web-сервіси, API, мультиагентна парадигма).

8. Загальні етапи побудови онтології

На основі аналізу існуючих методологій побудови онтологій та з урахуванням їх недоліків пропонуємо такі етапи побудови онтології:

1) вивчення й систематизування початкових умов. Ця дія встановлює основні цілі й контексти проекту розробки онтології, а також розподіляє ролі між членами проекту;

2) збір і накопичення даних. На цьому етапі відбувається збір і накопичення необхідних початкових даних для побудови онтології;

3) аналіз даних. Ця стадія полягає в аналізі і групуванні зібраних даних і призначена для полегшення побудови термінології;

4) початковий розвиток онтології. На цьому етапі на основі відібраних даних формується попередня онтологія. Цей етап складається з розробки прототипу структури, створення і нарощування робочої онтології, тестування і відображення та пов’язування з іншими онтологіями, інтеграція;

5) уточнення й затвердження онтології. Заключна стадія процесу. Використання, зберігання і підтримка онтологій;

6) експлуатація і технічне обслуговування. Розширення робочої онтології і повторне використання.

Зроблений аналіз інструментальних засобів підтримки процесу розробки онтологій дозволяє зробити висновок про те, що ці засоби ще досить незрілі і не пов’язані один з одним, що виключає інтероперабельність, і повинні розвиватися в напрямі підтримки інтероперабельності, зміни інтерфейсу на API, спрощування та розроблятися з підтримкою найбільш актуального на сьогодні напряму нового стандарту подання онтологій OWL 2.0.

Питання для самоконтролю

1. Охарактеризуйте онтології як засіб представлення знань у Web.
2. Опишіть формальну модель онтології і приклади її застосування.
3. Які основні характеристики онтології?

4. Яка логіка використовується як теоретичний базис для онтологічного представлення знань?
5. Які інструментальні засоби побудови онтологій?
6. Які існують методології розробки онтологій?
7. Які є методології розвитку онтологій?
8. Укажіть загальні етапи побудови онтології.

ТЕМА 6
ТЕХНОЛОГІЇ ТА СТАНДАРТИ SEMANTIC WEB

1. Головні задачі Semantic Web.

2. Компоненти Semantic Web.

3. Стандарти Resource Description Framework (RDF) і Web Ontology Language (OWL) – мов, що лежать в основі Semantic Web.

1. Головні задачі Semantic Web

Semantic Web пропонує потужний практичний підхід до отримання засобів керування великою кількістю інформації та інформаційних послуг. Врахування семантики інформаційних ресурсів надає засоби для того, щоб зробити інформацію більш корисною. Але це потребує нових засобів подання даних, які покращують здатність до збирання та спільного використання знань, а також і нових конструкцій програмування та інструментів, що дозволяють обробляти цю інформацію у прикладних програмах.

Головними задачами Semantic Web є створення мов семантичної розмітки електронних документів; словників, які описують структуру і семантику елементів семантичної розмітки, та реалізація засобів автоматичного генерування та обробки цієї семантичної інформації. У Semantic Web мають підтримуватися такі функції:

– індексація і пошук інформації;

– розробка та підтримка метаданих;

– розробка та підтримка методів анотування;

– подання Web у вигляді великої, інтероперабельної бази даних;

– організація машинного видобутку даних;

– виявлення (discovery) та надання Web-орієнтованих сервісів;

– дослідження в галузі інтелектуальних програмних агентів;

– розвиток підтримки інтерфейсів різних пристроїв та всепроникаючі обчислення.

2. Компоненти Semantic Web

Основними компонентами Semantic Web є онтології, сервіси та програмні агенти. Semantic Web базується на наборі відкритих стандартів:

- опис метаданих RDF;
- мова подання онтологій OWL;
- SPARQL – мова запитів до RDF;
- мова визначення правил SWRL.

Вплив Semantic Web формує нові перспективи в розробці програмного забезпечення, що базуються на «data-centric» моделі, яка використовує велику кількість різноманітних розподілених даних та має велику виразність, полегшене спільне використання та більшу гнучкість. Програмування в Semantic Web – це потужний новий підхід, що забезпечує більш ефективне використання великих обсягів інформації та робить доступними різноманітні сервіси. Модель знань Semantic Web може інтегруватися з прикладними програмами і використовуватися для окремих доменоорієнтованих бізнес-логік із самої програми. Таким чином, Semantic Web – це потужний напрям для підвищення ефективності розподіленого та сумісного доступу до інформації та її використання прикладними програмами.

Semantic Web можна уявити як Web даних, описаних і пов’язаних таким чином, щоб створити контекст або семантику, які представлені за допомогою чітко визначених граматичних та мовних конструкцій. Розширені прикладні програми Semantic Web можуть здійснювати інтеграцію словників предметної області (та їх онтології). Врахування семантики інформаційних ресурсів дозволяє більш ефективно використовувати наявні дані для вирішення проблем. Якщо семантика джерела інформації не формалізована, то це потребує від користувачів або складних програмних інструкцій певним чином виявляти її.

Базові компоненти, з яких складаються твердження Semantic Web, – це універсальний ідентифікатор ресурсу (URI), мови Semantic Web, онтології і екземпляри даних. Для реалізації Semantic Web потрібні відповідні програмні інструменти. Інструменти бувають

чотирьох типів: інструменти конструювання для створення і вдосконалення прикладних програм Semantic Web, інструменти запитів для дослідження Semantic Web, різонери (засоби логічного виведення), щоб додати виведення для Semantic Web, а також машини правил для розширення Semantic Web.

На відміну від інших програм прикладні програми Semantic Web зосереджені не на програмних інструкціях, а на даних. Багатство даних Semantic Web спрощує цю задачу. Це дозволяє відокремлювати дані від програмних інструкцій та дозволяє створювати більш гнучкі рішення. Крім того, прикладні програми Semantic Web є Web-орієнтованими: вони використовують переваги від масштабування, різноманітності і розподіленості WWW. Багато сучасних програм намагаються вирішити ці проблеми, але вони не в змозі повною мірою скористатися з WWW і працюють лише в обмеженому, ізольованому інформаційному просторі. Застосування Semantic Web здатні використовувати розмір і різноманітність WWW шляхом створення стандартизованої виразної інформації.

Semantic Web використовує набір нових відкритих інформаційних стандартів, які можуть застосовувати й всі інші. Прикладні програми, що базуються на цих стандартах, можуть швидко впроваджувати нові джерела інформації.

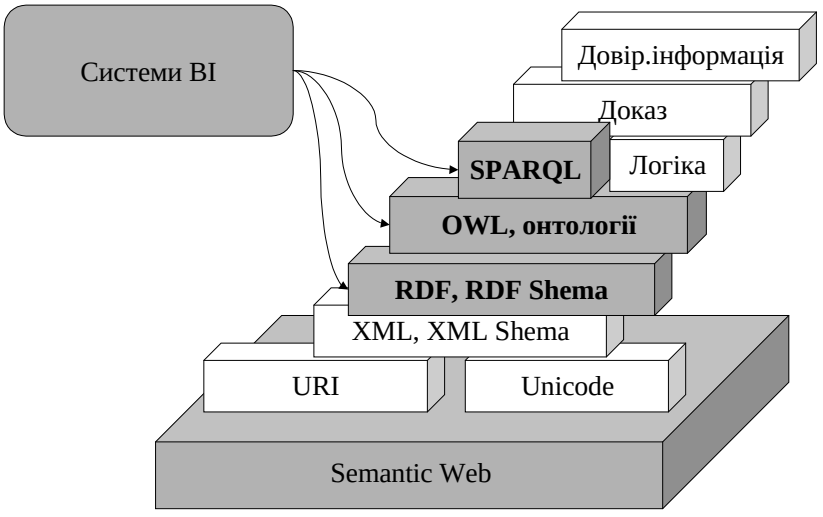

Рис. 6.1. Стек технологій Semantic Web

На сьогодні проект Semantic Web активно розвивається, з'являються нові мови та стандарти роботи з розподіленими знаннями й вдосконалюються існуючі. Тому доцільно у процесі розробки моделей, методів та засобів підтримки сервіс-орієнтованих прикладних програмних систем орієнтуватися саме на ці досягнення Semantic Web і створювати семантичні Web-сервіси, здатні ефективно використовувати всі переваги нового інформаційного середовища.

Semantic Web – це злиття Web-технологій і науки про представлення знань (knowledge representation, KR), що є підобластю штучного інтелекту (artificial intelligence, AI), спрямоване на створення і підтримку потенційно складних моделей світу, що дозволяють міркувати про себе і про пов'язану з ними інформацію.

3. Стандарти Resource Description Framework (RDF) і Web Ontology Language (OWL) – мов, що лежать в основі Semantic Web

Стандартизація Resource Description Framework (RDF) і Web Ontology Language (OWL) – мов, що лежать в основі Semantic Web, і визрівання нових технологій до вбудовування семантики в існуючі Web-сторінки і до запитів RDF до сховищ знань показують, що в цій області явно відбувається щось дуже важливе.

BI 2.0 забезпечує більш високий рівень абстракції і працює із семантичною моделлю даних. Це дозволяє здійснювати пошук потрібної користувачу інформації на семантичному рівні, використовуючи її метаопису. Для підтримки цього використовуються такі технології Semantic Web (рис. 3):

Resource Description Framework (RDF);

Web Ontology Language (OWL);

SPARQL (SQL like query language for RDF).

RDF

RDF (Resource Description Framework) – стандарт, прийнятий у 1999 році консорціумом W3C і підтриманий багатьма ведучими виробниками програмного забезпечення і постачальниками

контента, що являє собою модель опису метаданих. Він надає можливість формулювати твердження у виді, придатному для обробки комп'ютером.

Мета RDF – запропонувати базову модель даних «об'єкт – атрибут – значення» для метаданих. Крім цієї передбачуваної семантики, описаної у стандарті лише неформально, RDF не містить будь-яких чітких правил, орієнтованих на моделювання даних. Також як і XML Schema використовується для визначення словника, RDF Schema дозволяє розроблювачам визначати конкретний словник для даних RDF (такий, як authorOf) і указувати види об'єктів, до яких можуть застосовуватися ці атрибути. Іншими словами, механізм RDF Schema надає базову систему типів для моделей RDF. Ця система типів використовує деякі визначені терміни, такі як Class, subPropertyOf і subClassOf, для схеми, орієнтованої на конкретне застосування. Вираз схеми RDF також є коректним виразом RDF (як і вирази схеми XML – коректні вирази XML).

У BI 2.0 RDF використовується для створення метаописів ресурсів, які сумісно створюються та використовуються користувачами BI-платформ.

Наприклад, представники Oracle оголосили, що на сьогодні уже більше ніж у 100 комерційних системах і системах з відкритим кодом використовується реалізація технології Semantic Web, розроблена Oracle для СКБД Oracle Database 10g Enterprise Edition. Для них створюються Java-розширення, що забезпечують можливість збереження даних у Oracle 10g у форматі RDF.

Наприклад, включення компанією Oracle у липні 2005 року підтримки RDF у свій продукт Spatial 10.2g забезпечило легітимність RDF.

У ході експериментів з базами даних RDF у багатьох випадках проявилися їхні безсумнівні переваги перед традиційними структурованими базами даних, особливо у відношенні вбудовування даних у Web. Як відзначала компанія Microsoft у Services Framework 3.0 Developer Guide за 2006 рік, збереження профілів із застосуванням RDF вигідно через дві причини: 1) RDF забезпечує гнучку схему збереження даних, що дозволяє

зберігати застосовувані типи інформації, про які не було відомо при початковій розробці схеми; 2) допомагає створювати Web-подібні відношення між даними, що нелегко зробити в звичайній реляційній базі даних.

OWL

Онтологія OWL (Web Ontology Language) є послідовністю аксіом і фактів, а також посилань на інші онтології. Онтології також містять компоненти для запису авторства та іншої подібної інформації. Онтології OWL є документами Web, на них можна посилатися через URI. Мова подання онтологій OWL розширює можливості XML, RDF, RDF Schema та DAML+OIL.

Онтологія визначає терміни, за допомогою яких можна описати ПрО. Використання онтології особливо необхідно в застосуваннях-агентах, що здійснюють пошук і об'єднання інформації з різних джерел і з різних середовищ, у яких той самий термін може означати різні речі. Незважаючи на те, що формальний опис структури XML-документів DTD (Document Type Definition) цілком достатній для обміну даними між сторонами, що заздалегідь домовилися про значення визначень і термінів, відсутність семантики в зазначених засобах опису структури серйозно обмежує надійність виконання задачі пошуку й об'єднання даних при використанні нових XML-словників.

У BI 2.0 OWL використовується для інтегрованого подання знань відповідних ПрО.

SPARQL

З визнанням RDF назріла потреба у стандартній мові запитів для RDF, що відіграє ту ж роль, що і SQL у реляційних даних. Протокол SPARQL і мова запитів RDF Query Language розроблені саме з цією метою. Твердження SPARQL як стандартна мова запитів для RDF дозволяє багатьом сховищам даних стати у вигляді точок доступу SPARQL, у такий спосіб забезпечуючи гнучкий обмін даними між системами. Це надає шлях до Web-застосувань, що мають свого роду «фрактальну» структуру із шаблонами, у яких один компонент використовує інший як джерело даних (наприклад, через SPARQL) і сам виступає в ролі джерела даних для третього компонента.

SPARQL – мова запитів, розроблена для моделі даних RDF. Використання твердження SPARQL як стандартної мови запитів для RDF дозволяє багатьом сховищам даних стати точками доступу SPARQL, у такий спосіб забезпечуючи гнучкий обмін даними між системами. Це надає шлях до Web-застосувань, у яких один компонент використовує інший як джерело даних (наприклад, через SPARQL) і сам виступає в ролі джерела даних для третього компонента.

SPARQL одночасно є як мовою запитів, так і протоколом доступу до даних, одним із ключових компонентів Web 2.0: як стандарт для підтримки гнучкої моделі даних він надає спільний механізм запитів для Web 2.0. SPARQL використовується для представлення запитів до різноманітних джерел даних, незалежно від того, зберігаються ці дані безпосередньо в RDF або представляються у виді RDF за допомогою проміжного програмного забезпечення (middleware). Використання запитів SPARQL в BI 2.0 забезпечує інтегрований доступ до розподілених знань.

Питання для самоконтролю

1. Які головні задачі Semantic Web?
2. Назвіть компоненти Semantic Web?
3. Яке призначення стандартів Resource Description Framework (RDF) і Web Ontology Language (OWL) – мов, що лежать в основі Semantic Web?
4. Дайте приклад опису метаданих RDF?
5. Яке призначення мови подання онтологій OWL?
6. Яке призначення SPARQL – мови запитів до RDF?
7. Яке призначення мови визначення правил SWRL?

ТЕМА 7
DATA MINING ЯК ЗАСІБ ІНТЕЛЕКТУАЛІЗАЦІЇ ПОВЕДІНКИ ПРОГРАМНИХ АГЕНТІВ ТА МУЛЬТИАГЕНТНИХ СИСТЕМ

1. Основні властивості програмних агентів.
2. Теоретичні передумови виникнення програмних агентів.
3. Архітектури агентів.
4. Мультиагентні системи.
5. Мови комунікації агентів.
6. Засоби інтелектуалізації поведінки програмних агентів.

1. Основні властивості програмних агентів

Програмні агенти (далі – ПА) – нова парадигма програмування, яка дозволяє перейти на новий, більш інтелектуальний рівень взаємодії користувача з програмним і апаратним забезпеченням. Вона сприяє підвищенню ефективності праці та дозволяє користувачам доручити ІС виконання досить складних завдань. Значна частина з них є інтелектуальними і здатні здобувати, накопичувати та використовувати знання, використовуючи для цього методи та засоби Data Mining.

Задачі, які стають перед програмними агентами, досить часто є занадто складними, щоб вирішувати їх за допомогою одного агента, і це спонукало до створення нової концепції – мультиагентних систем (МАС).

Для того щоб ефективно використовувати ПА для рішення інтелектуальних задач, потрібно знати можливості ПА та теоретичні засади їх розробки.

2. Теоретичні передумови виникнення програмних агентів

Теорія ПА використовує результати, отримані у процесі досліджень інших наукових дисциплін: теорії керування, розподіленого штучного інтелекту та когнітивної психології.

Програмні агенти – програмні сутності, здатні діяти автономно та цілеспрямовано у динамічному середовищі для того, щоб виконати завдання користувачів.

Формальний опис динамічних систем через апарат теорії керування дозволяє прогнозувати їх поведінку. Якщо ця теорія допомагає досліджувати взаємодію агента і середовища, в якому він функціонує, то питання про те, як у агента виникають цілі і наміри і які механізми далі приводять до їхньої реалізації у вигляді дій, можна вирішувати за допомогою методів когнітивної психології і, зокрема, теорії мотивації. Основні напрями теорії мотивації:

– *процес формування наміру*: за множиною мотивацій, можливо, суперечливих, визначається результуюча мотиваційна тенденція, що стає базисом для формування наміру до дії;

– *процес формування бажань і дій:* виходячи зі сформованих на попередньому етапі намірів визначаються та ініціюються дії.

Ідея агентів запропонована у 50-х роках XX століття Мак-Картні та Селфриджем, які називали агентом програмну систему, що за наявності поставленої мети самостійно виконує певні операції і може при цьому запитувати й отримувати вказівки користувача, подані у термінах природної мови.

Агент – це насамперед комп’ютерна програма. З цього випливають такі властивості, як коректність, повнота, ефективність, надійність. При цьому агент виконує певні функції людини, надаючи користувачу потрібні йому послуги.

Вимога неперервності й автономії викликана потребою в тому, щоб агент був здатний гнучко реагувати на зміни середовища без постійного втручання користувача. Крім того, агент, який працює в середовищі з іншими агентами і процесами, має бути здатним спілкуватися з ними.

Термін «агентно-орієнтоване програмування» (АОП) був запропонований Шохамом для опису набору дій, необхідних для створення ПА – програмних сутностей, що функціонують безперервно й автономно в конкретному оточенні, у багатьох випадках – разом з іншими процесами й агентами. Можна розглядати агентний підхід як метафору проектування та моделювання розподілених систем.

Питання про те, яку комп’ютерну програму варто кваліфікувати як ПА, і досі не знайшло однозначного рішення. Різноманіття підходів і застосувань показує, що ПА стали одним з базових напрямів досліджень у галузі ІТ. Однак при цьому термін «програмний агент» використовувався без будь-якої спільної угоди про його значення. У результаті цього деякі програми стали називати агентами тільки тому, що вони, приміром, могли використовуватися для подання завдань віддаленим комп’ютерам або були здатні переміщатися самостійно між ними. Тому потрібно дати більш точне визначення агентам.

ПА – автономна фізична або віртуальна обчислювальна сутність, що базується на:

– власних ресурсах – знаннях та вміннях;

– засобах сприйняття середовища (сенсорах) та впливу на це середовище (ефекторах);

– моделі середовища, заснованої на знаннях про нього.

ПА забезпечують наступні функціональні можливості:

– вирішення задач або досягнення певних цілей на основі наявних ресурсів та навичок;

– вибір рішення між альтернативами та виконання цього рішення у певному середовищі;

– спрямована взаємодія з іншими агентами та середовищем, у якому функціонує ПА.

З погляду користувача, основна перевага використання агентів полягає в спрощенні взаємодії з програмою – користувачу досить поставити загальну задачу перед агентом, не вдаючись у подробиці того, як саме агент має її вирішувати. Якщо агент має можливості для рішення цієї задачі, він вирішує її сам, а інакше запитує необхідні йому послуги в інших агентів.

Агент – це програмний об’єкт, який:

– забезпечує виконання однієї або кількох корисних послуг;

– надає опис семантики цих послуг іншим ПА;

– здатний функціонувати автономно без безпосередніх вказівок користувача;

– може інтерактивно взаємодіяти з іншими ПА та користувачами.

Найпростіше визначення агента ґрунтується на *моделі чорного ящика*, що знаходиться у певному середовищі: агент описується як функція f, що обробляє інформацію від сенсорів і вхідні повідомлення (рис. 7.1), а результат роботи агента – його дії і вихідні повідомлення.

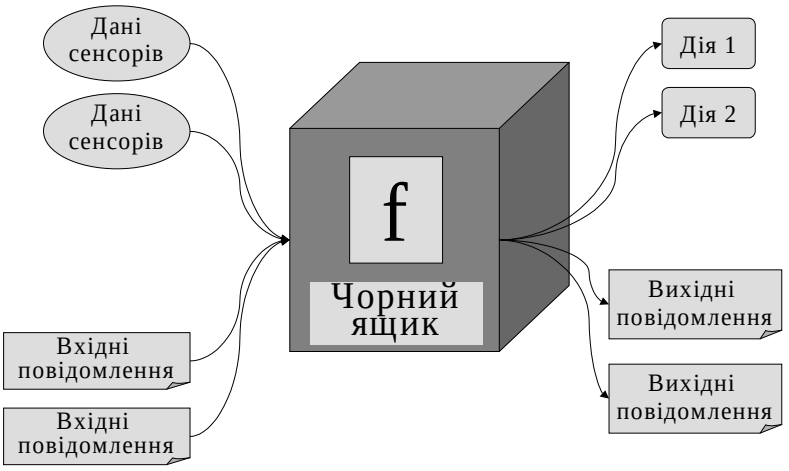

Рис. 7.1. Визначення агента через модель чорного ящика

За визначенням FIPA (Federation of Intelligent Physical Agents), *агент* – це об’єкт, що знаходиться в певному середовищі, від якого він отримує дані про події в цьому середовищі, інтерпретує їх і виконує команди, що впливають на середовище. Такий агент може містити як програмні, так і апаратні компоненти. FIPA – це міжнародна організація, створена в 1996 році з метою впровадження агентної парадигми для розробки практичних застосувань.

Можна визначати ПА через множину його атрибутів. ПА – термін, що дозволяє об’єднати множину більш специфічних і обмежених типів агентів, які мають деякі з таких атрибутів:

– *реактивність* (reactivity) – зміна своєї поведінки залежно від конкретної ситуації;

– *автономність* (autonomy) – самостійне виконання розпоряджень користувача без детальних інструкцій;

– *співробітництво* collaborative behavior) – здатність працювати разом з іншими агентами для досягнення спільної мети;

– *спілкування на рівні знань* («knowledge level» communication ability) – спроможність спілкуватися з людьми й іншими агентами мовою, близькою до природної;

– *здатність до логічного виведення* (inferential capability) – обробка абстрактного опису задачі з використанням апріорних знань про цілі і найбільш придатні методи їх досягнення, спроможність будувати моделі власної сутності, свого користувача, ситуацій та інших агентів;

– *безперервність у часі* (temporal continuity) – стійкість ідентифікації і положення протягом тривалого періоду часу;

– *персоналізація* (personality) – наявність персоналізованих значень атрибутів власної поведінки;

– *адаптивність* (adaptivity) – навчання й удосконалення на основі власного досвіду;

– *мобільність* (mobility) – здатність самостійно переходити з однієї платформи на іншу.

– *правдивість* – припущення про те, що агент не буде свідомо поширювати помилкову інформацію;

– *лояльність* – намагання робити те, що потрібно іншим агентам;

– *раціональність* – виконання тільки тих дій, які приводять до досягнення цілей.

Однією з основних характеристик агента є комунікабельність – здатність до гнучкого спілкування як з агентами, так і з іншими програмними компонентами.

Підкласом ПА є інтелектуальні агенти (ІА), які мають засоби для збереження знань (бази знань – БЗ) та механізми для їх обробки (рис. 7.2).

ІА у широкому розумінні має такі ключові ознаки:

автономність (autonomy) – функціонування значною мірою незалежно від втручання людини і контроль власних дій та внутрішнього стану;

соціальність (social ability) – інтелектуальна та конструктивна взаємодія з іншими агентами та людьми шляхом обміну з ними повідомленнями деякою загальнозрозумілою мовою комунікацій;

реактивність (reactivity) – сприйняття зміни середовища і вчасне реагування на них;

проактивність (pro-activity) – здатність агента генерувати цілі і діяти раціонально для їх досягнення, а не тільки реагувати на зовнішні події.

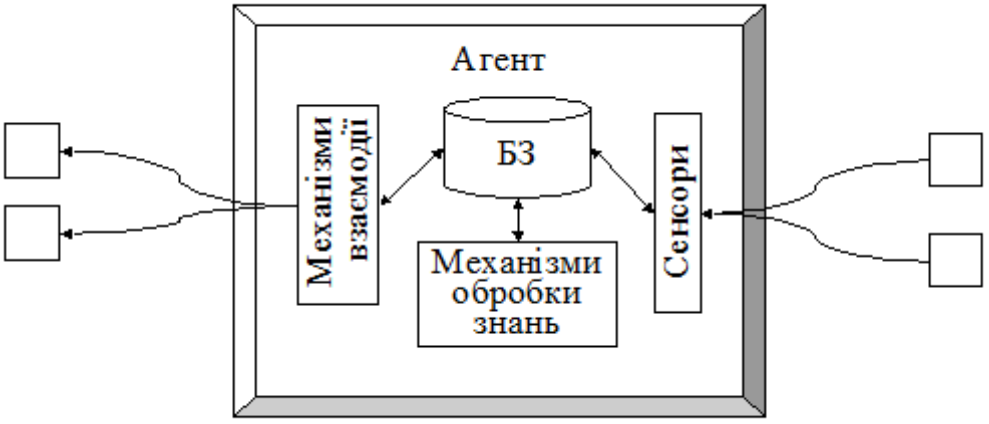

Рис. 7.2. Загальна концептуальна схема інтелектуального ПА

Інтелектуальність ПА визначається його спроможністю міркувати і навчатися, яка зазвичай базується саме на Data Mining, наявністю моделі користувача, його потреб і механізму пошуку засобів їх задоволення.

Вирізняють такі ознаки інтелектуальності агентів:

– автономне виконання своїх функцій;

– взаємодія з іншими агентами і користувачами;

– здатність стежити за оточенням;

– використання абстракції;

– застосування знання ПрО;

– адаптивність поведінки;

– навчання на власному досвіді;

– толерантність до помилок у вхідних сигналах;

– здатність працювати в реальному часі;

– спілкування природною мовою.

Використання агентів сприяє розвитку принципово нових інформаційних технологій.

Модель ПА містить модель ПрО, модель користувача, засоби сприйняття (сенсори), засоби виконання дій (ефектори), цілі і планувальник дій на підставі цілей, моделі інших агентів і засоби взаємодії з ними.

Модель ПрО, у якій функціонує ПА, відображає структуру та ієрархію об’єктів (наприклад, у вигляді онтології), на які спрямовані дії агентів та які впливають на способи досягнення його цілей. ПА має явно задану символьну модель світу, у якій рішення (наприклад, вибір дії) приймаються через логічні (або щонайменше псевдологічні) міркування, що базуються на відповідності між зразками та символьними маніпуляціями. Це приводить до двох проблем: як за час, упродовж якого інформація ще буде актуальною, адекватно описати реальний світ символами та як агентам обробити цю інформацію.

Модель користувача призначена для того, щоб ПА правильно інтерпретував завдання користувача та сповіщав про отримані результати у зручній та зрозумілій формі. У процесі роботи агент може поповнювати модель користувача, накопичуючи досвід взаємодії з конкретним користувачем (або класом користувачів) для підвищення ефективності своєї роботи.

Засоби сприйняття та засоби виконання дій ПА залежать від функцій, які агент має виконувати. Цілі агента визначають його дії відповідно до принципу раціональності: агент виконує тільки ті дії, які за наявною в нього інформацією та правилами її обробки приведуть до виконання хоча б однієї з його цілей. Планувальник дій агента визначає його дії та їх послідовність.

Моделі інших агентів потрібні ПА для того, щоб успішно взаємодіяти з цими агентами й обмінюватися з ними інформацією у процесі досягнення спільних цілей.

Таксономії програмних агентів

3. Архітектури агентів

Основні принципи побудови та функціонування ПА називають їх *архітектурою*. Залежно від того, які принципи визначають дії агентів, архітектури поділяються на деліберативні (агенти

обирають план дії на основі логічного виведення з наявних в них знань) та реактивні (дії агентів визначаються як реакція на події у зовнішньому середовищі).

На практиці зазвичай застосовують різноманітні комбінації цих архітектур, які називають *гібридними*. Крім того, деякі дослідники виділяють в окремі класи архітектури з певними спільними рисами (приміром, інтерактивні архітектури, архітектури з планувальником дій, архітектури інтелектуальних агентів, інтенсіональні архітектури). Архітектура ПА відображає внутрішню організацію та взаємодію між основними компонентами (рис. 7.3).

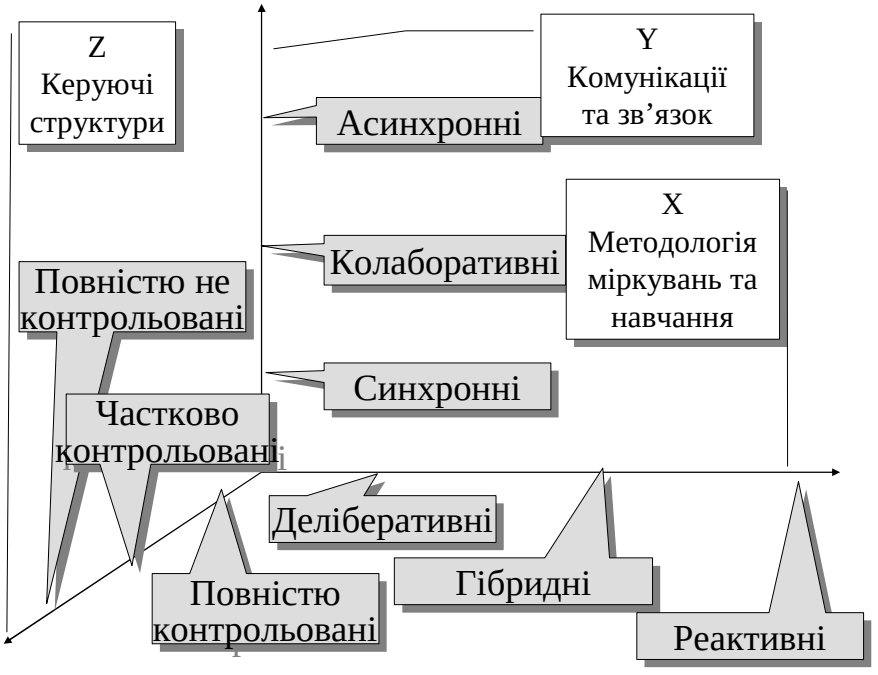

Рис. 7.3. Таксономія архітектур мобільних агентів

Деліберативна архітектура ПА містить символьну модель світу, подану у явній формі, за допомогою якої ПА на основі міркувань логічного чи псевдологічного типу приймає рішення про дії, які він здійснює. Такий агент може розглядатися як спеціальний випадок системи, заснованої на знаннях. Використання деліберативної архітектури ПА приводить до кількох принципових проблем:

– обсяг символьної інформації, яку зберігає ПА, пропорційний розміру внутрішнього стану, тому його збільшення призводить до зниження мобільності ПА;

– символьне подання інформації про середовище, на основі якої міркують ПА, має формуватися у режимі реального часу, щоб знайдені агентами рішення були корисними;

– перетворення сценаріїв реального світу на точне та адекватне символьне визначення є нетривіальною задачею.

Реактивні архітектури не використовують централізовану символьну модель світу та не застосовують складні символьні міркування. Така архітектура оперує на низькому рівні абстракції. Невеликий час очікування відповіді забезпечує ефективну взаємодію між агентами з такою архітектурою один з одним та з середовищем. Недолік архітектури – неможливість глибокого аналізу даних від сенсорів.

Реактивна архітектура забезпечує прийняття рішень за значно меншим обсягом інформації про навколишнє середовище та за значно менший час, використовуючи прості емпіричні правила, специфічні для певної ПрО.

Здатний міркувати ПА має явно подану символьну модель світу, у якій рішення (наприклад, про те, яку дію виконувати) продукуються через логічні (або, щонайменше, псевдологічні) міркування, засновані на відповідності зразків і символьних маніпуляцій.

З іншого боку, архітектури агентів класифікують залежно від типу структури функціональних компонентів ПА та методів організації взаємодії між ними. Як правило, архітектура агента організується у вигляді кількох рівнів.

Тільки найпростіші ПА можуть бути реалізовані за одноріневою схемою. Рівні відображають функції ПА, такі, як сприйняття зовнішніх подій і прості реакції на них; координація взаємодії з іншими ПА; відновлення переконань про зовнішній світ; визначення своїх дій на черговому кроці тощо. Найчастіше в архітектурі ПА присутні рівні, що відповідають за такі функції:

- сприйняття і виконання дій;
- реактивну поведінку;
- локальне планування;
- кооперативну поведінку;

- моделювання ПрО;
- формування намірів;
- навчання.

Гібридні архітектури. Як видно з наведених вище прикладів, як реактивний, так і деліберативний підхід мають певні переваги та недоліки. Тому доцільно поєднати ці підходи та розробити ПА, який містить дві підсистеми: 1) здатну міркувати, обробляти символьну модель світу, розробляти план і приймати рішення; 2) здатну реагувати на події, що відбуваються в навколишньому середовищі без проведення складних міркувань.

4. Мультиагентні системи

Проблеми, для вирішення яких застосовують агентний підхід, можуть бути досить складними. Дослідження в галузі ШІ показали, що взаємодія і декомпозиція загальної задачі ефективно позначається на результатах виконання цієї задачі. Тому краще створювати окремі автономні блоки такої системи, а потім організовувати їх спільне функціонування.

Термін *«мультиагентні системи»* використовується для позначення ІС, які складаються з множини автономних модулів ПА та мають такі властивості:

– кожен ПА є автономним, мобільним та інтероперабельним;

– ПА, що входять до складу МАС, здатні обмінюватися інформацією для досягнення спільних цілей;

– керування ПА може бути децентралізованим;

– джерела даних і доступ до них децентралізовані;

– робота агентів є асинхронною.

Теорія МАС походить від теорії відкритих систем, розподіленого ШІ і загальної теорії складних систем. Розподілений штучний інтелект (РШІ) пов’язаний з аналізом систем, що складаються з окремих незалежних об’єктів, які взаємодіють один з одним, та механізмів їх координації. МАС теж є предметом розгляду РШІ. У цьому випадку незалежними об’єктами є ПА. Для вивчення поведінки МАС використовують методи таких наукових дисциплін:

– *розподілений ШІ* (теорія розподілених систем, теорії прийняття рішень), що займається найбільш загальними аспектами колективної поведінки агентів;

– *теорія ігор*, яка використовується для дослідження ситуацій, аналогічних до кооперативних ігор, стратегій ведення переговорів;

– *теорія колективної поведінки автоматів*, яка досліджує колективну поведінку великих груп автоматів з примітивними функціями, спроможних навчатися за допомогою системи штрафів і заохочень;

– біологічні, економічні та соціальні моделі.

У розробці МАС використовуються також результати з інших галузей теоретичних досліджень (рис. 7.4).

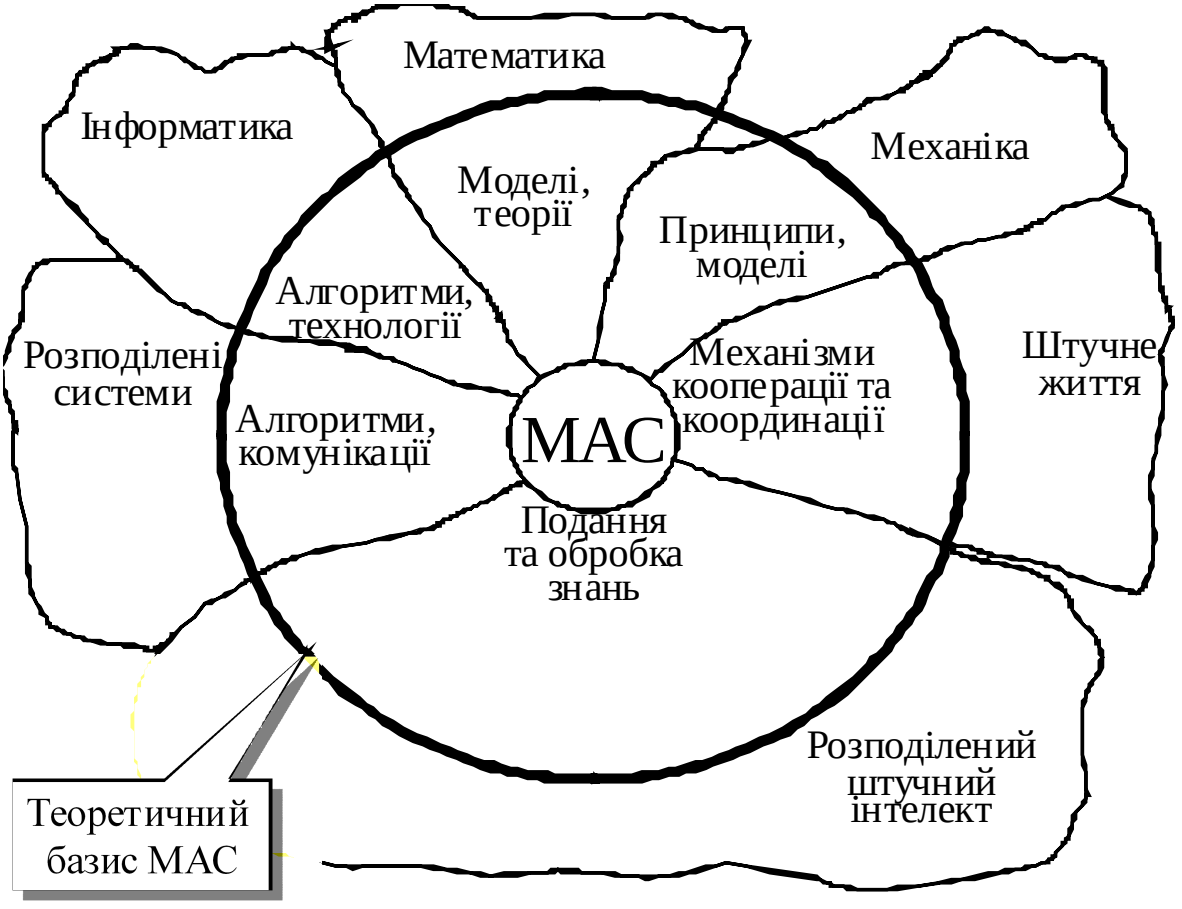

Рис. 7.4. Основні ПрО, що використовуються у розробці МАС

МАС – співтовариство ПА, які пов’язані один з одним цілями та ресурсами. ПА, що входять до складу МАС, здатні взаємодіяти для того, щоб обмінюватися послугами, які потрібні їм для досягнення цілей, поставлених перед ними користувачами. Агент, не здатний самостійно вирішити задачу, поставлену перед ним користувачем, звертається до інших агентів, які можуть надати йому відповідні послуги.

МАС поширені в багатьох ПрО: керуванні виробничими процесами і промисловими підприємствами; плануванні рухом транспорту (повітряного, залізничного, автомобільного); аналізі та пошуку інформації; навчанні; бізнесі тощо.

МАС складається з множини ПА, які:

– взаємодіють шляхом комунікацій;

– здатні діяти у певному середовищі;

– мають певні сфери впливу, які можуть перетинатися або співпадати.

Функціонування МАС пов’язане з кооперацією та конкуренцією агентів у процесі колективного вирішення задач. Агент, що не може вирішити власну задачу самостійно, має взаємодіяти з іншими ПА. При цьому агенти можуть будувати плани спільних дій, ґрунтуючись не тільки на власних можливостях, але і аналізувати плани та наміри інших ПА (використовуючи різні комбінації інтенсіональних відношень). Кооперативні дії ПА – важлива перевага МАС. При такій організації роботи група ПА проявляє новий, ефективніший тип поведінки.

Мотивація використання МАС базується на їх властивостях:

– здатність вирішувати проблеми, складні для одного централізованого ПА через обмеженість ресурсів;

– можливість взаємодії та інтероперабельності з різноманітними застосунками (ЕС, СППР тощо);

– вирішення дійсно розподілених проблем (приміром, керування рухом транспорту);

– знаходження рішень на основі розподілених ІР;

– вирішення проблем з розподіленою експертизою (приміром, медичний догляд);

– підвищення швидкості та надійності;

– здатність збільшувати кількість обчислювальних пристроїв (процесорів), що використовуються для виконання задачі;

– можливість опрацювання нечітких та неповних даних і знань.

Знання й уміння МАС здобуваються від великої кількості відносно простих ПА, що поєднуються разом деякою архітектурою. У процесі колективної роботи агенти мають будувати плани дій, ґрунтуючись не тільки на своїх можливостях, але й враховуючи плани і наміри інших агентів (на основі своїх знань та переконань щодо цих планів та намірів).

Причини взаємодії ПА:

– наявність сумісних цілей;

– нестача індивідуальних ресурсів ПА для досягнення цілей;

– нездатність ПА самостійно вирішити задачу;

– наявність взаємних зобов’язань.

Перехід до відкритих МАС надає можливість переходу на нову якість функціонування системи в силу того, що система (група агентів) більша ніж сума властивостей її членів (агентів).

Найпростіший метод координації МАС – *організаційне структурування* – полягає у чітко визначених і довгострокових відношеннях між ПА. При цьому використовують ієрархічні структури *master-slave* або *client-server*. Організаційне структурування передбачає, що принаймні один ПА має глобальне уявлення про всі співтовариства, однак для багатьох ПрО це не реально.

Цей метод використовується у двох варіантах:

– головний ПА планує і розподіляє завдання між підлеглими ПА, які, на відміну від головного, мають часткову автономність;

– для координації використовується дошка оголошень для обміну інформацією між ПА, операції з якою (зчитування й запис) визначає головний ПА.

Інший варіант координації – *укладення контракту* – припускає децентралізовану структуру. ПА мають дві ролі:

– менеджер поділяє задачу на підзадачі і шукає виконавця, щоб виконати їх;

– виконавець реалізує свою підзадачу.

Виконавець може рекурсивно стати менеджером і розбити свою підзадачу на ще дрібніші задачі і доручити їх іншим ПА.

Пошук виконавця складається з таких етапів:

– менеджер повідомляє задачу;

– виконавці оцінюють цю задачу щодо можливості її виконати;

– менеджер отримує таблицю виконавців, оцінює отримані пропозиції, обирає виконавця і доручає йому виконання певної задачі;

– виконавець виконує отриману задачу та повідомляє менеджера про отримані результати.

Така модель координації забезпечує розподіл задачі і засоби для самоорганізації співтовариства агентів. Її доцільно застосовувати у таких ситуаціях:

- задача має чіткий ієрархічний характер;
- задачу можна поділити на великі підзадачі;
- взаємозв’язок між підзадачами – відносно невеликий.

Переваги цього підходу координації полягають у динамічному розподілі задачі завдяки пошуку оптимального виконавця; збалансованому завантаженні ПА і надійному механізмі для розподіленого керування.

Існує два типи планування діяльності МАС:

- централізоване;
- розподілене.

У *централізованому* плануванні діяльності МАС є координаційний ПА, що отримує індивідуальні плани від інших, аналізує їх, щоб знайти потенційні протиріччя та конфлікти у взаємодії ПА. Потім координаційний ПА намагається змінити ці індивідуальні плани і поєднує їх у план МАС, у якому усунені суперечливі взаємодії та додані команди зв’язку, що синхронізують взаємодію ПА у потенційно можливих конфліктах.

У *розподіленому* плануванні діяльності МАС кожен ПА отримує моделі планів інших ПА, що входять до складу МАС. ПА взаємодіють, щоб побудувати модифікації своїх індивідуальних планів, які не конфліктують з планами інших ПА. Координація в

розподіленому плануванні діяльності МАС набагато складніша, ніж у централізованому випадку, тому що жоден ПА не володіє глобальним уявленням про всю розподілену систему.

Переговори – один з найскладніших методів координації, який використовує методи ШІ, логіку тощо. У динамічному співтоваристві агенти самостійно розподіляють роботи між собою. Часто одну і ту ж роботу в співтоваристві можуть виконати кілька ПА. Агенти на момент розподілу робіт знаходяться в різних станах (наприклад, мають різний ступінь завантаженості). Для ефективного розподілу робіт між ПА вони мають взаємодіяти один з одним – вести переговори, у процесі яких виявити оптимального виконавця для кожної роботи.

Процес переговорів складається з таких компонентів:

– протоколу переговорів – набору правил, за якими відбувається взаємодія агентів (учасники переговорів, їх типи, стани переговорів, правила, за якими змінюються стани переговорів, можливі дії учасників тощо);

– об’єкта переговорів – діапазону проблем, відносно яких потрібно досягти згоди;

– моделі ухвалення рішення ПА – апарат ухвалення рішення, що використовують учасники відповідно до протоколу переговорів.

Отже, при побудові МАС, що вирішує реальну задачу, як правило, необхідно використовувати комбінацію описаних вище підходів. При моделюванні діяльності віртуальних і реальних підприємств переважно застосовують організаційне структурування, яке задовільно відображає ієрархічну структуру підприємства. Крім того, це один з найпростіших підходів до координації. Інші методи координації мають серйозні недоліки при вирішенні задач подібного роду: переговори – через складність реалізації; планування МАС придатне лише для специфічних задач, таких, як планування авіарейсів. Укладення контракту при моделюванні віртуальних підприємств не надає переваг порівняно з організаційним структуруванням, тому що мережа контрактів фактично визначена до початку роботи і тому немає необхідності у пошуку виконавця.

Для МАС характерні децентралізованість даних, асинхронність обчислень і наявність засобів комунікації. Впровадження МАС дозволяє вирішувати проблеми, що є надто складними для окремого ПА або пов’язані з обмеженими ресурсами, та забезпечує збільшення ефективності системи через паралельні обчислення та повторне використання ПА в різних співтовариствах агентів. Для успішного функціонування вони потребують великих обсягів знань, які мають постійно оновлюватися та перевірятися. Це досягається шляхом обміну знаннями між ПА.

МАС можуть бути як централізованими, так і децентралізованими. У централізованих МАС, створених для вирішення агентами спільних задач, конфлікти між цілями агентів не виникають (або ж існують централізовані механізми вирішення таких протиріч). Наприклад, в інформаційно-пошуковій МАС, що обслуговує групу користувачів, пріоритети користувачів визначають порядок їх обслуговування. Проте такі МАС значно важче адаптувати до нових потреб користувачів або нових завдань.

Значна частина МАС, створених на сьогодні, складається з ПА, які не є повністю автономними (тобто здатними вибирати для себе цілі і вирішувати, яким чином досягати ці цілі та які саме дії виконувати для цього). Набір ПА розробляється цілісно, і проблема взаємодій між агентами вирішується у процесі їхньої розробки. Такі проблеми, як конфлікти або недостатність ресурсів, узагалі не розглядаються. Агенти, що входять до складу такої МАС, мають обмежену автономію: їхня роль у процесі рішення загальної проблеми звичайно заздалегідь визначається розробником системи, але агенти вільні у виборі способів досягнення своїх цілей.

Існує багато різноманітних типів МАС, але можна виділити характеристики, яким задовольняють деякі з цих систем:

– агенти, що входять до складу МАС, автономні;

– агенти поєднують свої індивідуальні потреби з потребами всієї системи.

МАС, що відповідає цим умовам, називають *відповідальним співтовариством.* Його компонентами є:

– *елементи*, призначені для рішення певних проблем;

– *середовище*, у якому знаходяться ці елементи;

– *способи взаємодії* між елементами;

– *цілі* – мотиваційна сила, під впливом якої елементи вирішують проблеми.

Разом ці компоненти характеризують систему.

5. Мови комунікації агентів

ПА можна розглядати як застосунки, для яких здатність до комунікації з іншими застосунками і до пошуку знань має першочергову важливість. Компоненти ПА можна поділити на такі: подання, комунікацій і ті, що не належать безпосередньо до розподіленої взаємодії.

Для рішення проблеми комунікацій між інтелектуальними ПА необхідна спільна мова, тобто спільні синтаксис, семантика й прагматика. Зараз не існує загальноприйнятої мови подання інформації і запитів. Такі мови, як KIF і SQL, досить поширені, але цього недостатньо, щоб прийняти їх як стандарт.

Агенти мають взаємодіяти на рівні знань і тому не можуть задовільно обслуговуватися звичайними мовами та протоколами, що призначені для розподілених обчислень і звичайно фокусуються на процесах, а не на програмах або коаліціях ПА. Мова комунікації агентів має бути досить потужною для підтримки взаємодії між програмами високого рівня. Мова комунікації – не тільки протокол.

Протокол взаємодії – це високорівнева стратегія, що здійснюється агентом і взаємодіє з іншими агентами. Такий протокол може знаходитися в діапазоні від схеми переговорів і протоколів теорії ігор до простих протоколів типу «*щораз, коли я щось не знаю, я знаходжу кого-небудь, хто знає, і я запитую його*». Вибір транспортного протоколу залежить від специфіки його застосування.

Можна сформулювати такі вимоги до мови взаємодії агентів:

– *форма*: декларативна, синтаксично проста, зрозуміла;

– *зміст*: відрізняє мови опису комунікаційних актів від мов передачі вмісту повідомлень;

– *семантика*: подає бажані властивості;

– *реалізація*: проста, сумісна з існуючим ПЗ;

– *робота в мережі*: враховує основні аспекти сучасних ІТ і має незалежний транспортний механізм;

– *середовище*: підтримує гетерогенність і динамізм;

– *надійність*: забезпечує надійність і безпеку взаємодії агентів.

Ці вимоги можуть конфліктувати одна з одною. Приміром, проста для розуміння людиною мова може бути не такою стислою, як можливо. Вибір вдалого співвідношення усіх вимог являє собою для розробника мови нетривіальну задачу.

Крім того, мова комунікацій має забезпечувати надійність, конфіденційність і безпеку взаємодії агентів на основі механізму ідентифікації агентів і виявлення помилок.

Для конструктивної й інтелектуальної взаємодії між ПА необхідна не тільки загальна мова, але й загальне розуміння термінів, що використовуються у конкретному контексті.

Таким чином, ПА взаємодіють у трьох напрямах:

– спільної мови;

– спільного розуміння знань, якими вони обмінюються;

– здатності обмінюватися знаннями за допомогою спільної мови.

Мови комунікацій і їх властивості

Вимоги до мов комунікації агентів можна поділити на групи: до форми, змісту, виконання, роботи в мережі, середовища, надійності. Для рішення проблем спілкування між агентами у рамках проекту KSE з метою створення мови для підтримки різноманітних архітектур агентів розроблено протокол KQML. При цьому особливий інтерес викликає спеціальний клас ПА – *посередники* (facilitators) комунікацій. Посередники – це агенти, що реалізують різноманітні комунікаційні сервіси, приміром, підтримують реєстрацію імен сервісів, переправляють повідомлення для них, виконують маршрутизацію повідомлень на підставі вмісту і забезпечують проміжний сервіс.

Подання знань ПА може бути поділено на дві підзадачі:

– переклад з однієї мови подання на іншу;

– пошук семантичного контексту подання знань для різних застосувань.

Комунікації викликають проблеми, які пов’язані з протоколом взаємодії, мовою комунікацій, транспортним протоколом.

Крім протоколів, ПА можуть містити інші компоненти, що допомагають виконувати свої задачі. Здатності міркувати про свої дії, представляти метазнання, планувати дії або моделювати інших агентів дозволяють підвищити можливості застосувань. Такі компоненти є периферійними для бази знань. Вони також можуть використовувати базу знань і комунікаційну мову.

KSE – це ініціативна група з розробки і розвитку технічної інфраструктури для підтримки пошуку знань між системами. KSE було організовано три робочі групи, кожна з яких вирішувала одну з проблем технології подання знань: Interlingua, Shared Reusable Knowledge Bases, External Interfaces. Група *Intrelingua* розробила загальну мову подання вмісту БЗ. Цією групою був опублікований документ, що описує формат обміну знаннями KIF (Knowledge Intrchange Format). KIF може бути використано для забезпечення перекладу з однієї мови на іншу або як загальну мову для двох агентів, що використовують різні мови подання (http://www.cs.umbc.edu/kse/kif/). Група *SRKB* (Shared Reusable Knowledge Bases) займалася розробкою розподілених БЗ і пошуком знань за визначеними темами, а також розробкою предметно-незалежних інструментів і методологій. Група *External Interfaces* розглядала питання взаємодії систем, заснованих на базах знань.

KIF є версією числення предикатів першого порядку з доданням підтримки немонотонних міркувань і визначень. Визначення мови включає опис синтаксису і семантики. KIF містить логічні оператори, які допомагають подавати логічну інформацію (заперечення, диз’юнкція, правила, квантифікаційні формули тощо). KIF дозволяє кодувати знання про знання, використовуючи символи «‘» і «,», оператори і пов’язаний покажчик. KIF можна також використовувати для опису процедур, тобто програм ПА. Семантика KIF (без правил і визначень) подібна логіці першого порядку. Вона розширюється за рахунок використання нестандартних операторів і обмежується аксіомами моделювання.

За винятком цих розширень і обмежень ядро мови зберігає фундаментальні характеристики логіки першого порядку.

KQML – мова і набір протоколів, що забезпечують ПА ідентифікацію, зв'язок і обмін інформацією з іншими агентами.

Основні риси KQML:

- повідомлення KQML непрозорі щодо вмісту;
- перформативи (performatives) – примітиви мови – визначають припустимі операції, які може здійснити ПА для комунікацій з іншими агентами;
- середовище агентів, що використовують KQML, може бути поповнене спеціальними агентами для виконання сервісних функцій: зв'язку фізичних адрес із символьними іменами, реєстрації БД, комунікаційних послуг тощо.

KQML складається з трьох рівнів:

- вмісту;
- комунікацій;
- повідомлень.

Рівень *вмісту* – це фактичний зміст повідомлення власною мовою подання (KQML може переносити висловлювання будь-якою мовою подання, у тому числі у вигляді рядків ASCII). Деякі ПА, що використовують KQML (приміром, програми маршрутизації, узагальнені посередники), можуть ігнорувати вміст повідомлення і тільки визначати, що воно закінчується.

Рівень *комунікацій* містить набір властивостей, що описують низькорівневі параметри комунікації (такі, як ідентифікація відправника й одержувача, ідентифікатор комунікації тощо).

Рівень *повідомлень* використовується для кодування того, що одна програма намагається передати іншій. Цей рівень формує ядро мови KQML і визначає вид взаємодії з агентом, що використовує KQML. Основною функцією рівня повідомлень є ідентифікація протоколу, що використовується для передачі повідомлення і забезпечення виконання певної дії, яку відправник зв'язує з повідомленням (запиту, команди тощо). Крім того, на цьому рівні можуть бути описані властивості мови вмісту, визначатися його тематика. Це дозволяє KQML виконувати аналіз, маршрутизацію і доставку повідомлень навіть у тому випадку, якщо їхній вміст недоступний.

Синтаксис KQML ґрунтується на збалансованому списку з дужками. Початковим елементом списку є перформатив, а інші елементи є його аргументами у вигляді пар *(ключове слово, значення).*

6. Засоби інтелектуалізації поведінки програмних агентів

Для досягнення успіху в складному динамічному середовищі інтелектуальний ПА потребує засобів адекватного реагування на зміни в своєму оточенні або невідповідність апріорних припущень про це оточення. Поведінка агента описується *принципом раціональності*: він обирає дію, якщо має знання про те, що ця дія приведе до однієї з його цілей. Агенти, що мають однакові знання і цілі, на рівні знань нерозрізнені, навіть якщо вони мають різну фізичну структуру.

Рівень інтелектуальності ПА визначається його здатністю до міркування і навчання, тобто можливостями здобуття необхідних знань за доступною інформацією. Критерії оцінювання самостійно здобутих знань збільшують автономність агента, а вміння ПА враховувати свій досвід спілкування з користувачами підвищує його ефективність. Тому засоби здобуття знань мають бути невід'ємною складовою моделей інтелектуальних ПА.

Система автономна (у розумінні «не знаходиться під безпосереднім контролем людини») у тій мірі, у який її поведінка визначається її власним досвідом.

Агент, що планує свої дії на основі вбудованих припущень, діє успішно тільки в тому випадку, якщо ці припущення виконуються, але цим припущенням може бракувати гнучкості. Внаслідок динамічності оточення ПА використання вбудованих припущень часто призводить до неефективних дій. Автономний ПА успішно діє навіть при великій розмаїтості оточення, маючи достатній час на адаптацію. Для цього він використовує елемент навчання, який за допомогою зворотного зв'язку порівнює переконання агента про доцільність виконання доступних дій з реальними результатами виконання цих дій. Якщо знаходяться розбіжності, то елемент навчання намагається внести зміни в переконання ПА. У процесі розробки елемента навчання треба враховувати, які переконання ПА потрібно удосконалити, як вони подані, як здійснюється зворотний зв'язок та яка апріорна інформація доступна ПА.

Можливість навчання є важливою властивістю будь-якої системи, яка функціонує у середовищі, що динамічно змінюється. У МАС проблеми навчання особливо актуальні, але мають ряд особливостей. При цьому інтелектуальність розглядається вже не як властивість окремого ПА, а як властивість групи взаємодіючих агентів. Тому необхідно розглядати такі питання:

- у чиїх інтересах має здійснюватися навчання (окремого ПА, групи агентів, усієї МАС);
- як здійснюється процес навчання (самостійно, через взаємодію з іншими ПА, усією групою);
- що є об’єктом навчання (поведінка ПА, взаємодія тощо);
- який метод навчання використовується;
- як перевіряти ефективність навчання.

Інтелектуальна поведінка системи пов’язана зі здатністю класифікувати об’єкти (а саме задача класифікації є найпоширенішою в Data Mining). Проблема, яку має вирішити ПА, полягає в тому, щоб за набором прикладів (з власного досвіду або досвіду інших ПА, з якими він здатний взаємодіяти) побудувати гіпотезу, яка узагальнює ці приклади або дозволяє знаходити рішення в аналогічних ситуаціях. Задача прогнозування властивостей полягає у визначенні невідомих характеристик об’єкта дослідження.

Використання механізмів навчання на базі прецедентів у МАС дозволяє забезпечити адаптивність їх поведінки. Механізми навчання можуть бути застосовані для вибору моделі поведінки агента, стратегії переговорів, протоколів взаємодії, прийнятті рішень тощо. Це пов’язано з використанням різноманітних методів ШІ: логіки, виведення на основі фактів, цілеспрямованого пошуку тощо. Розрізняють такі напрями:

– розподілене виведення на основі фактів;

– колективне виведення на основі фактів;

– проактивне навчання;

– здобуття знань на основі аукціонів – координоване здобуття прецедентів у МАС.

Якщо навчається один ПА з МАС, то він самостійно вирішує, чи є певний прецедент корисним для його навчання. У МАС

приклад, який не потрібний одному ПА, може бути корисним для іншого. Тому такий процес навчання складається з двох етапів:

- акумуляції прецедентів;
- обміну прецедентами.

Для етапу акумуляції прецедентів виділяють три стратегії:

- агент ніколи на додає прецедент до своєї бази фактів;
- агент завжди додає прецедент до своєї бази фактів;
- агент додає прецедент до своєї бази фактів, якщо він є цікавим (приміром, якщо ПА неправильно класифікує ситуацію, описану у прецеденті).

На етапі обміну прецедентами визначають стратегію поведінки ПА. Приміром, агент ніколи не пропонує прецедент для навчання іншим агентам або завжди пропонує його іншим ПА.

Агент A_i може отримати від агента A_j опис прецедентів, які можуть допомогти йому у вирішенні певної задачі, за таким протоколом:

A_i надсилає опис проблеми P іншим агентам $A_j, j=\overline{1,n}$, що входять до складу МАС;

A_j здійснює спробу вирішити проблему, використовуючи свою базу прецедентів C_j;

A_j надсилає A_i такі повідомлення:

відмову, якщо він не може вирішити проблему;

рішення у вигляді трійки $\langle S_k, P, A_j \rangle$, яке означає, що агент A_j вважає S_k найбільш ймовірним вирішенням проблеми P.

Схема голосування визначає механізм, за допомогою якого агент A_i будує узагальнене рішення на основі рішень, що надійшли від інших агентів $A_j, j=\overline{1,n}$. Приміром, узагальнене рішення може визначатися як клас, що отримав найбільшу кількість голосів (комітетна політика).

Висновки

Агентна парадигма привносить ряд принципово нових властивостей і можливостей в інформаційні технології і являє собою якісно новий рівень її розвитку, який забезпечується розподіленими

обчисленнями в гетерогенному інформаційному середовищі Інтернету. Впровадження агентних технологій має підвищити ефективність інтелектуальної діяльності людини, позбавивши її від рутинних операцій. Одним з чинників інтересу до МАС став розвиток мережі Інтернету, в якій функціонують розподілені автономні програмні системи, що обробляють гетерогенні інформаційні ресурси. Використання Data Mining забезпечує інтелектуальність поведінки ПА та надає їм можливості до навчання за власним досвідом.

Делегування складних задач інтелектуальним мультиагентним системам дозволяє представляти і вирішувати проблеми, які важко формалізуються. Можна виділити такі основні напрями застосування інтелектуальних, знання-орієнтованих агентних технологій:

- пошук, фільтрація та аналіз інформації;
- ефективний доступ до Web-сервісів;
- ситуаційне керування;
- дистанційне навчання;
- електронний бізнес;
- моніторинг даних;
- телемедицина;
- керування виробництвом у режимі on-line;
- керування транспортними потоками.

Питання для самоконтролю

1. Що таке програмні агенти?
2. Які властивості мають програмні агенти?
3. Що таке FIPA?
4. Які атрибути мають програмні агенти?
5. Що таке проактивність програмного агента?
6. Як класифікують архітектури програмних агентів?
7. Для чого потрібні мультиагентні системи?
8. З яких компонентів складається процес переговорів між агентами?

ТЕМА 8
ІНТЕЛЕКТУАЛЬНІ WEB-СЕРВІСИ

1. Сервіс-орієнтована архітектура.
2. Концепція Web-сервісів.
3. Онтологічний опис Web-сервісів.
4. Профіль сервісу.

1. Сервіс-орієнтована архітектура

Сервіс-орієнтована архітектура (далі – СОА) – це концепція проектування, розробки й керування функціональних модулів (сервісів), кожний з яких доступний через мережу і здатний виконувати певні дії.

СОА створює комунікаційне середовище для модулів, що реалізують прикладну бізнес-логіку. Інформація про модулі публікується в такій формі, що їх використання не вимагає знань про використані в них рішення і технології. Розробнику не потрібно знати, як працює програма, необхідно лише розуміти, які вхідні і вихідні дані потрібні, і як викликаються ці програми для виконання.

Сервіс-орієнтовані обчислення – це обчислювальна парадигма, яка використовує сервіси як фундаментальні елементи для розробки застосувань. Вони базуються на СОА і забезпечують виконання операцій керування сервісами. Розробка таких систем – це процес пошуку, дослідження та композиції сервісів, що задовольняють вимогам користувача.

Можливість композиції сервісів часто розглядають як одну з основних переваг їх використання. Вона полягає у знаходженні набору елементарних сервісів, необхідних для реалізації функцій, використовуваних у запиті користувача, і визначення порядку їх виконання.

SOA – це архітектурний шаблон програмного забезпечення, модульний підхід до розробки програмного забезпечення, заснований на використанні розподілених, слабко пов’язаних замінних

компонентів, оснащених стандартизованими інтерфейсами для взаємодії за стандартизованими протоколами.

Технологія SOA для керування бізнес-процесами є великим кроком вперед з точки зору підвищення ефективності розробки систем; за значимістю її можна порівняти із створенням у кінці 50-х років компіляторів мови високого рівня. Цей підхід дозволяє спростити використання Web-сервісів з будь-якого місця розташування і їх виконання на основі бізнес-правил.

SOA підштовхує до використання альтернативних технологій і підходів (таких як обмін повідомленнями) для побудови застосувань за допомогою зв'язування сервісів, а не за допомогою написання нового програмного коду. У цьому випадку, при належному проектуванні, застосування повідомлень дозволяє компаніям своєчасно реагувати на зміну ринкових умов – настроювати процес обміну повідомленнями, а не розробляти нові програми.

SOA – це термін, який з'явився для опису виконуваних компонентів – таких як Web-сервіси – які можуть викликатися іншими програмами, які виступають як клієнти або споживачі цих сервісів.

У найзагальнішому вигляді SOA припускає наявність трьох основних учасників:

– постачальника сервісу;

– споживача сервісу;

– реєстру сервісів.

Взаємодія учасників виглядає досить просто: постачальник сервісу реєструє свої сервіси в реєстрі, а споживач звертається до реєстру із запитом.

Для використання сервісу необхідно дотримуватися угоди про інтерфейс для звернення до сервісу – інтерфейс повинен не залежати від платформи. SOA реалізує масштабованість сервісів – можливість додавання сервісів, а також їх модернізацію. Постачальник сервісу і його споживач виявляються непов'язаними – вони спілкуються за допомогою повідомлень. Оскільки інтерфейс повинен не залежати від платформи, то і технологія, використовувана для визначення повідомлень, також повинна не залежати від платформи. Тому, як правило, повідомлення є XML-документами, які відповідають XML-схемі.

2. Концепція Web-сервісів

Концепція Web-сервісів виникла наприкінці 90-х років XX ст. і стала галузевим стандартом у сфері ІКТ. Стандарти Web-сервісів розроблені такими компаніями, як IBM, Microsoft, Ariba, Sun Microsystems, SAP за підтримки Консорціуму W3C. У межах W3C було створено робочу групу Web Services Architecture Working Group, яка опублікувала глосарій термінів у сфері Web-сервісів.

Web-сервіси використовують XML для обміну даними між застосуваннями, незалежно від використання операційної системи, апаратної платформи і розробника. Web-сервіс – це набір логічно пов'язаних функцій, які можуть бути програмно викликані через мережу Internet. Web-сервіс – це програма, що ідентифікується через URI, інтерфейс якої може бути подано у вигляді мови XML.

Web-сервіси – це реалізована програмними засобами система для підтримки міжкомп'ютерної взаємодії телекомунікаційних мереж, що підтримується такими стандартами: SOAP (Simple Object Access Protocol) – протокол обміну повідомленнями; WSDL – мова опису програмних інтерфейсів Web-сервісів; UDDI (Universal Description, Discovery and Integration) – класифікатор Web-сервісів.

Інформація про те, які функції пропонує конкретний Web-сервіс, міститься в його описі – WSDL-документі. Інші системи взаємодіють з Web-послугами, використовуючи повідомлення у стандарті за протоколом SOAP, передані з використанням HTTP і XML і в поєднанні з іншими Web-стандартами.

Для пошуку Web-сервісів використовують спеціальні реєстри, що підтримують UDDI. Є два основні методи публікації Web-сервісів для користувачів – UDDI і DISCO. UDDI – це централізований структурований сервіс реєстру, а DISCO пропонує вільну форму механізму пошуку через браузер. Реєстр UDDI – центральне сховище для специфікацій та інформації про підприємства, включаючи послуги, які компанії надають шляхом Internet.

SOAP

Web-сервіси стають доступними через протоколи HTTP GET, HTTP POST, HTTP SOAP.

SOAP – стандарт передачі повідомлень через Internet, розроблений фірмою Microsoft для віддаленого виклику процедур (RPC, Remote Procйdure Call) через протокол HTTP. Він дає змогу передавати інформацію мережею у форматі XML. Можуть використовуватися будь-яка мережа, будь-який протокол передачі даних, довільна інформація, різні обчислювальні пристрої (зокрема мобільні). Специфікація SOAP визначає XML–«конверт» для передачі повідомлень, метод для кодування програмних структур даних у форматі XML, а також засоби зв'язку через протокол HTTP.

WSDL

WSDL – заснований на XML стандарт опису того, як користуватися сервісом, запропонований Консорціумом W3C. Опис Web-сервісу мовою WSDL містить технічні деталі, необхідні для інтеграції Web-сервісу у застосування (формат повідомлень, операції). На сьогодні WSDL підтримують продукт від Microsoft – SOAP Toolkit 2.0 (WSDL Generator) і продукт від IBM – WSDL Toolkit. Мова опису Web-сервісів (Web Services Description Language (WSDL)) визначає синтаксис того, як Web-сервіс може бути викликаний.

UDDI

Стандарт UDDI надає механізм виявлення Web-сервісів. UDDI формує бізнес-реєстр (UDDI Business Registry), в якому провайдери Web-послуг можуть реєструвати свої послуги, а розробники – відшуковувати необхідні їм сервіси. Компанії реєструють себе в Business Registry, який є базою даних загального користування. UDDI дає можливість описувати, інтегрувати і публікувати сервіси. UDDI сам є спеціалізованим Web-сервісом, що дає змогу користувачам і застосуванням знаходити необхідні їм сервіси.

Інтелектуальний пошук та автоматичне компонування Web-сервісів можуть бути здійснені за допомогою можливостей семантичного опису Web-сервісів, запропонованих у OWLS.

3. Онтологічний опис Web-сервісів

Семантичні Web-сервіси

Інтелектуальні, або *семантичні* Web-сервіси – елементи програмної логіки з однозначно описаною семантикою, доступні через Інтернет і придатні для автоматичного пошуку, композиції і виконання з урахуванням їх семантики. Вони розширюють поняття звичайних Web-сервісів.

Для опису семантики Web-сервісів недостатньо звичайних засобів подання відомостей про Web-сервіс (приміром, стандарт WSDL, UDDI), хоча WSDL, приміром, допускає наявність у цих описах довільного застосованого XML-вмісту, але необхідно розробити для опису Web-сервісів семантичних мов розмітки, подібних тим, що розробляє Консорціум W3C для статичної частини Web: RDF, RDF Schema, OWL тощо (рис. 8.1).

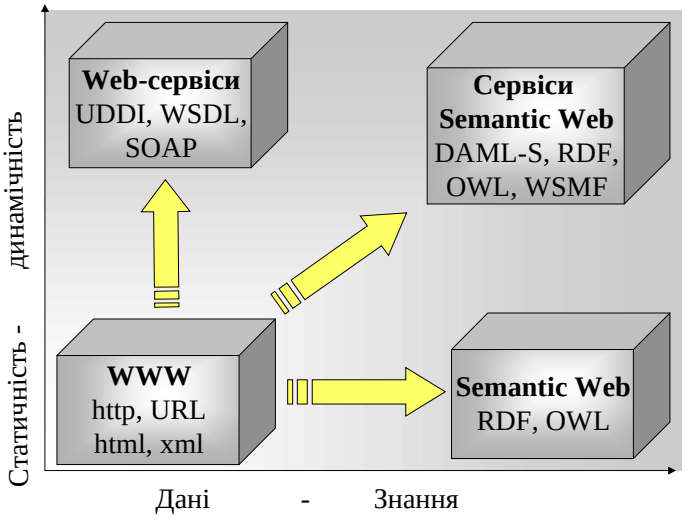

Рис. 8.1. Процес інтелектуалізації Web-сервісів

OWL-S – мова семантичного опису Web-сервісів

OWL-S забезпечує онтологічний опис Web-сервісів. Мета розробки OWL-S полягає в тому, щоб зробити можливим використання логічного виведення для Web-сервісів, планування автоматичного компонування Web-сервісів, автоматичного використання сервісів програмними агентами.

OWL-S забезпечує декларативні описи властивостей Web-послуги і можливості, які можуть використовуватися для автоматичного виявлення сервісу.

Використовуючи OWL-S, Web-сервіс може повідомляти потенційним користувачам про свої функціональні можливості. Запит на обслуговування може бути узгоджений з оголошенням Web-сервісів за допомогою процесу підбору (matchmaking).

OWL-S забезпечує механізм для моделювання бізнес-процесів, але відрізняється від нього виразністю термінів, уявлень, семантики, підтримки пошуку і виконання, обробки помилок. Опис OWL-S для сервісу складається з профілю сервісу, моделі сервісу та обґрунтування сервісу, тобто пояснення того, що виконує цей сервіс, як він працює, як можна дістати до нього доступ.

4. Профіль сервісу

OWL-S дозволяє описувати Web-сервіс у термінах *профілю*, що визначає, що саме робить сервіс; *моделі процесу*, що описує, як сервіс працює, і *бази (Grounding)*, що визначає, як одержати доступ до сервісу. Профіль і модель процесу є абстрактними специфікаціями в тому розумінні, що вони не описують у деталях окремі формати, протоколи і мережні адреси, з якими зв’язаний Web-сервіс. Для забезпечення цих конкретних деталей призначена підстава. Мова опису Web-сервісів WSDL, розроблена безпосередньо від OWL-S, надає для цього цілком задовільні засоби. Тому автори OWL-S вирішили використовувати його як основу для OWL-S. Передбачається, що поняття бази в OWL-S є розвитком поняття зв’язування (*binding*) у WSDL. У результаті основною задачею стає створення бази для атомарних процесів OWL-S.

Профіль сервісу – абстрактна характеристика функцій сервісу. Профіль сервісу будується на основі контенту UDDI, що описує властивості сервісу, необхідні для його автоматичного виявлення, такі, наприклад, як його входи і виходи, попередні умови і застосовувані дії. На основі профілю, що представляє інформацію про провайдера, функціональні можливості і функціональні атрибути сервісу, можуть бути створені описи і запити сервісу. Профіль Web-сервісу дає опис його властивостей: категорію сервісу (наприклад, за класифікацією UNSPSC) і його якісну оцінку (швидкість, надійність та ін.).

Для семантичного обґрунтування параметрів Web-сервісів використовують онтології різного рівня. Використовують різну архітектуру для опису семантики джерел інформації: підходи, засновані на єдиній онтології, використовують одну глобальну онтологію, що забезпечує спільний словник для специфікації семантики; підходи, засновані на множинних онтологіях, кожне джерело інформації описується власною онтологією; гібридні підходи подібні підходам, заснованим на множинних онтологіях у тому, що семантика кожного початкового тексту описана її власною онтологією, але для того, щоб зробити локальні онтології порівнянними, формується глобальний словник для загального використовування.

Розмітка Web-сервісів за допомогою OWL-S полегшує використання Web-сервісу: його автоматизоване виявлення Web-сервісу, його виконання, взаємодію, компонування і контроль за виконанням.

Щоб використовувати Web-сервіс, потрібно інтерпретувати комп’ютером визначення сервісу і засоби доступу до нього.

Для інтероперабельного подання онтологічного опису сервісів розроблено модифікацію OWL – OWL-S OWL-S забезпечують онтологічний опис Web-сервісу. Ціль розробки OWL-S полягає в тому, щоб уможливити використання логічного виведення для Web-сервісів, планування компонування Web-сервісів, автоматичне використання сервісів програмними агентами. OWL-S забезпечує декларативні описи властивостей Web-сервісу і можливості, що можуть використовуватися для автоматичного виявлення сервісу.

OWL-S – онтологічна мова, призначена для опису Web-сервісів. Ця специфікація розроблена W3C на основі DAML-S і пропонує структуру мови, що базується на іншій розробці концерну W3C – мові подання онтологій OWL. OWL-S має допомогти користувачам і пошуковим агентам виявляти, викликати, компонувати і контролювати Web-сервіси. OWL-S – це онтологія сервісів, призначена для того, щоб допомогти користувачам і пошуковим агентам виявляти, викликати, компонувати і контролювати Web-сервіси.

OWL-S (Web Ontology Language for Services) – важливий крок у напряму до більш інтелектуальних Web-сервісів, яка забезпечує онтологічний опис Web-сервісу. OWL-S дозволяє використовувати логічне виведення для Web-сервісів, планувати компонування Web-сервісів та автоматично викликати сервіси програмними агентами. OWL-S забезпечує декларативні описи властивостей Web-сервісу і можливості, що можуть використовуватися для автоматичного виявлення сервісу.

Для інтероперабельного опису сервісів розроблено модифікацію OWL для OWL-S (Web Ontology Language for Services). OWL-S забезпечують онтологічний опис Web-сервісу.

Ціль розробки OWL-S полягає в тому, щоб уможливити використання логічного виведення для Web-сервісів, планування компонування Web-сервісів, автоматичне використання сервісів програмними агентами. OWL-S забезпечує декларативні описи властивостей Web-сервісу і можливості, що можуть використовуватися для автоматичного виявлення сервісу.

Щоб використовувати Web-сервіс, програмні агенти мають потребу у визначенні сервісу, що інтерпретується комп’ютером, і засобів доступу до цього визначення. OWL-S – це онтологія сервісів, призначена для того, щоб допомогти користувачам і пошуковим агентам виявляти, викликати, компонувати і контролювати Web-сервіси.

Щоб використовувати Web-сервіс, потрібно автоматизовано інтерпретувати визначення сервісу і засоби доступу до нього. OWL-S – це онтологія сервісів, що реалізує цю функціональність. Поштовхом до його розробки став проект Semantic Web, що прагне забезпечити доступ до інформаційних ресурсів Web не за ключовими словами, а за контентом. Одна з цілей Semantic Web – створити засоби, що дозволяють автоматично користувачам вибирати, використовувати, компонувати і відслідковувати Web-сервіси. Важливим результатом у цьому напрямі стала розробка мов розмітки Web, таких як OWL і його попередник DAML+OIL. Ці мови дозволяють створювати онтології кожної предметної області (ПрО) і встановлювати зв’язки з цими онтологіями Web-сайтів для їхнього опису.

Сервіси можуть бути *простими* у тому розумінні, що вони активізують тільки якусь одну доступну через Інтернет програму, сенсор чи пристрій, не використовуючи інші Web-сервіси, і тому взаємодія між користувачем і сервісом, крім простого виклику, відсутня. Сервіси можуть бути і *комплексними*, що складаються з кількох простих. Вони звичайно вимагають взаємодії чи діалогу між користувачем і сервісами, тобто користувач може здійснювати вибір або накладати певні умови.

OWL-S призначений для підтримки обох категорій сервісів, але комплексні (складені) сервіси пред'являють більше вимог до властивостей мови.

Кожна доступна через Web програма, сенсор чи пристрій, що оголошені сервісом, можуть розглядатися як сервіс. OWL-S не виключає оголошення сервісом простих, статичних Web-сторінок. Однак основною метою OWL-S є підтримка більш складних задач.

Структура онтології OWL-S

Онтології OWL-S структурує відомості про сервіси на три категорії (рис. 8.2):

Профіль сервісу: клас SERVICE представляє SERVICEPROFILE;

Модель сервісу: клас SERVICE описується в SERVICEMODEL;

База сервісу: клас SERVICE підтримується в SERVICEGROUNDING.

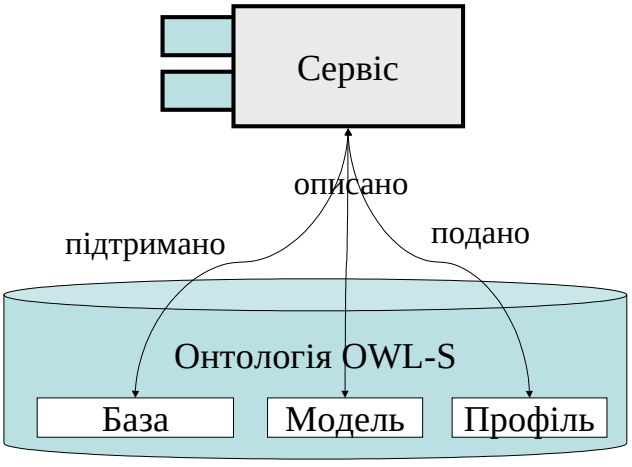

Рис. 8.2. Подання онтології Web-сервісу в OWL-S

Для кожного опублікованого сервісу має існувати один екземпляр класу SERVICE. Використовуючи OWL-S, Web-сервіс може повідомляти про свої функціональні можливості потенційним користувачам. Запит на обслуговування може бути погоджений з оголошеннями Web-сервісів за допомогою процесу підбору (matchmaking). OWL-S забезпечує механізм для моделювання бізнесів-процесів, але відрізняється від нього за виразністю термінів, подань, семантики, підтримки пошуку і виконання, обробки помилок.

Задачі OWL-S

OWL-S дозволяє вирішити задачі чотирьох типів:

1. Автоматичне виявлення Web-сервісів містить у собі автоматичне визначення місцезнаходження Web-сервісів, що надають певну послугу і задовольняють накладеним обмеженням. Наприклад, користувач хоче знайти дистанційний курс «Системи штучного інтелекту», у якому навчальні матеріали подані російською мовою, кількість годин не перевищує 60, а сам курс надається безкоштовно. Якщо сервіс розмічений за допомогою OWL-S, тоді семантична інформація, необхідна для знаходження сервісу, подана на сайті Web-сервісу у формі, що інтерпретується комп’ютером, і тоді пошукова система, здатна обробляти онтологічну інформацію, може знайти такий сервіс автоматично.

2. Автоматичний виклик Web-сервісу полягає в автоматичному виконанні ідентифікованого раніше Web-сервісу комп’ютерною програмою чи агентом. Наприклад, сервіс реєструє користувача як слухача певного дистанційного курсу і надає йому доступ до інтерактивної лекції, а потім викликає модуль-тестування. Виконання Web-сервісу можна розглядати як послідовність викликів функцій. Розмітка OWL-S Web-сервісу повинна забезпечувати декларативний, інтерпретований комп’ютером API для виконання цих функціональних викликів. Програмний агент повинний бути здатний інтерпретувати розмітку, щоб зрозуміти, які вхідні дані потрібні для виклику сервісу, яка інформація буде повернута і як виконати сервіс автоматично.

Автоматична композиція і взаємодія Web-сервісів містить у собі автоматичний пошук, композицію і взаємодію Web-сервісів

для виконання визначеної задачі, обумовленої високорівневим описом завдання. Наприклад, при дистанційному навчанні треба правильно визначити порядок досліджуваних дисциплін (від більш простих до більш складного), а тестування повинне викликатися після вивчення курсу.

Автоматичний моніторинг виконання Web-сервісів дозволяє з'ясовувати, на якому етапі знаходиться процес виконання запиту і чи не виникли будь-які непередбачені перешкоди. Наприклад, користувач системи дистанційного навчання може одержувати інформацію про свої поточні оцінки, про те, скільки годин потребує завершення навчання, які нові курси з'явилися за час його навчання та ін.

Для семантичного обґрунтування параметрів Web-сервісів використовують онтології різного рівня: онтології застосування, онтології ПрО й онтології верхнього рівня. Доцільно використовувати загальний словник (тезаурус), що містить базові терміни ПрО, використовувані в онтологіях застосувань (рис. 8.3). Кожне поняття в такому тезаурусі може розумітися через набір інших понять, що приводить до появи семантичного поля.

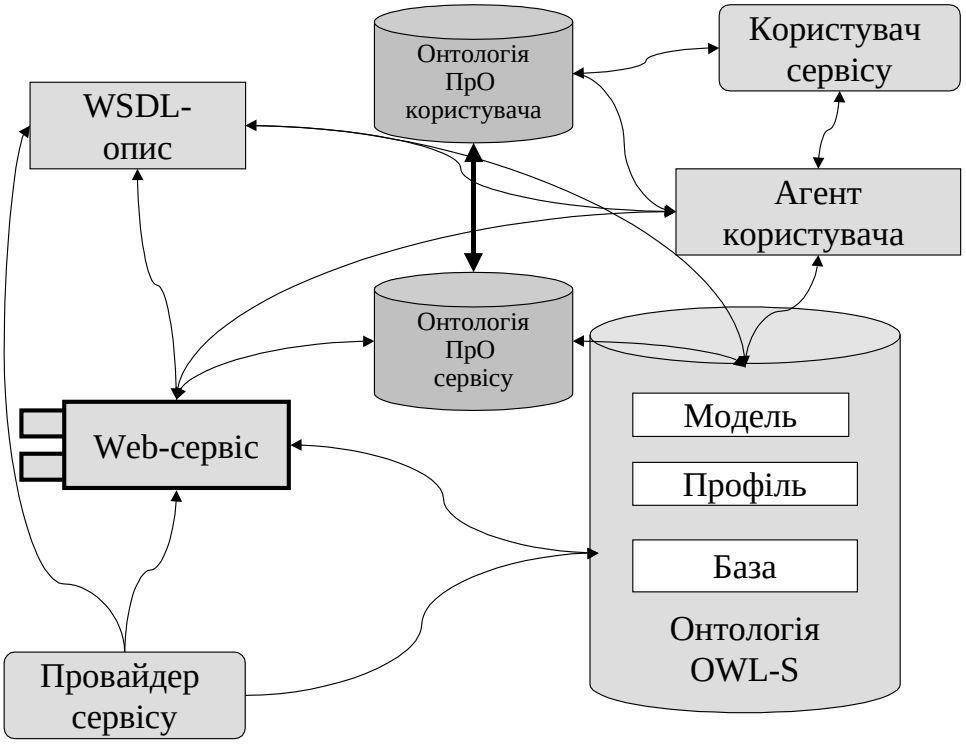

Рис. 8.3. Засоби подання семантики Web-сервісу

Створення профілю сервісу: відношення з моделлю процесу. Профіль сервісу забезпечує коротке визначення зареєстрованого сервісу. Але після того, як користувач звертається до сервісу, він використовує модель процесу для керування взаємодією із сервісом. Профіль та модель – це два різні подання одного й того ж сервісу.

OWL-S не нав’язує будь-які обмеження на моделі профілів та процесів, тобто обидва визначення можуть бути несумісними без впливу значущості виразу OWL.

Висновки

Онтологія OWL-S забезпечує механізм для моделювання бізнес-процесів, але відрізняється від нього виразністю термінів, уявлень, семантики, підтримки пошуку і виконання, обробки помилок. Опис OWL-S для сервісу складається з профілю сервісу, моделі сервісу та обґрунтування сервісу, тобто пояснення того, що виконує цей сервіс, як він працює, як можна дістати до нього доступ. Саме ця структура може використовуватися як основа для інтелектуального аналізу відомостей щодо сервісів та застосування методів Data Mining з метою автоматизованої компоновки сервісів відповідно до задач, сформульованих користувачами.

Питання для самоконтролю

1. Що таке Web-сервіси?
2. Як в SOAP використовується XML?
3. Що містить опис Web-сервісу мовою WSDL?
4. Для чого призначається UDDI Business Registry?
5. Чим відрізняються семантичні Web-сервіси ?
6. Що таке профіль сервісу в OWL-S?
7. Що таке OWL-S?

ТЕМА 9
НЕЙРОННІ МЕРЕЖІ

1. Сутність та види нейронних мереж.

2. Процес побудови і використання нейромережевих моделей:

- Призначення і підготовка навчальної вибірки.
- Налаштування призначення полів.
- Нормалізація значень полів.
- Приклади нормалізації полів.
- Налаштування навчальної вибірки.
- Налаштування структури нейромережі.
- Алгоритми навчання нейромережі.
- Карти Кохонена.

3. Гібридна модель нейронної мережі та її реалізація в пакеті Fuzzy Logic Toolbox системи MATLAB. ANFIS-редактор.

1. Сутність та види нейронних мереж

Нейронні мережі (НС) є обчислювальними структурами, моделюючими прості біологічні процеси, аналогічні процесам, що відбуваються в людському мозку.

Нейромережі – це розподілені і паралельні системи, здатні до адаптивного навчання шляхом реакції на позитивні і негативні дії.

Концептуальною основою і складовою частиною штучних нейронних мереж є *елементарний перетворювач*, званий *штучним нейроном, який має* визначену внутрішню структуру (рис. 9.1) і правила перетворення сигналів (6.1).

Штучний нейрон (далі просто – нейрон) складається з помножувачів (синапсів), суматора і нелінійного перетворювача. *Синапси*, зображені перекресленим кружком, призначені для зв'язку нейронів між собою і множать вхідний сигнал x_i на деяке постійне число w_i. Це число w_i, назване *вагою синапса*, характеризує *силу* цього зв'язку. *Суматор* виконує додавання всіх сигналів,

що надходять на вхід нейрона від інших нейронів, і зовнішніх вхідних сигналів. *Нелінійний перетворювач* призначений для нелінійної зміни вихідного значення суматора відповідно до деякої функції від одного аргументу. Ця функція називається *функцією активації* чи *передатною функцією* нейрона.

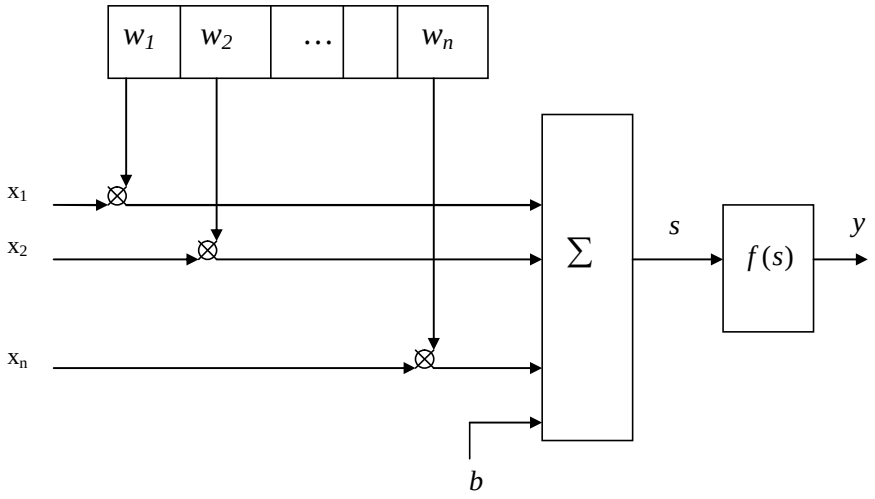

Рис. 9.1. Структура штучного нейрона

Правила перетворення сигналів визначаються математичною моделлю нейрона, що може бути записана у формі таких аналітичних виразів:

$$s = \sum_{i=1}^{n} w_i \cdot x_i + b; \qquad y = f(s), \tag{9.1}$$

де w_i – вага синапса $(i \in \{1,2,\ldots,n\})$; b – значення зсуву; s – результат підсумовування; x_i – компонент вектора чи входу вхідного сигналу $(i \in \{1,2,\ldots,n\})$; y – вихідний сигнал нейрона; n – число входів нейрона; *f(s)* – функція активації (передатна функція) нейрона, що представляє собою деяке нелінійне перетворення. У загальному випадку: $w_i, x_i, b \in \Re (i \in \{1,2,\ldots,n\})$.

Синаптичні зв'язки з позитивними вагами: $w_i \in \Re_+ (i \in \{1,2,\ldots,n\})$ називаються *збуджуючими*, а з негативними вагами $w_i \in \Re_- (i \in \{1,2,\ldots,n\})$ – *гальмуючими.*

Таким чином, окремо взятий штучний нейрон цілком описується своєю структурою (рис. 9.1) і математичною моделлю (6.1). Одержавши вектор вхідного сигналу x_i, нейрон видає деяке число *y* на своєму виході.

Як функція активації нейрона, можуть бути використані різні нелінійні перетворення (табл. 9.1).

Таблиця 9.1

Основні види функцій активації нейронів		
Назва	Область значень	Формула
Лінійна	$\Re$	$f(s) = k \cdot s$
Напівлінійна	$\Re_+$	$f(s) = \begin{cases} 0, \text{ якщо } s \le 0 \\ k \cdot s, \text{ якщо } s > 0 \end{cases}$
Гранична	$\{0, 1\}$	$f(s) = \begin{cases} 0, \text{ якщо } s < T \\ 1, \text{ якщо } s \ge T \end{cases}$
Модульна	$\Re_+$	$f(s) = \lvert s \rvert$
Знакова (сигнатурна)	$\{-1, 1\}$	$f(s) = \begin{cases} -1, \text{ якщо } s \le T \\ 1, \text{ якщо } s > T \end{cases}$
Квадратична	$\Re_+$	$f(s) = s^2$
Експоненціальна	$\Re_+$	$f(s) = e^{-a \cdot s}$
Синусоїдальна	[-1, 1]	$f(s) = \sin(s)$
Логістична (сигмоїдальна)	(0, 1)	$f(s) = \dfrac{1}{1 + e^{-a \cdot s}}$
Раціональна (сигмоїдальна)	(-1, 1)	$f(s) = \dfrac{s}{a + \lvert s \rvert}$
Гіперболічний тангенс (сигмоїдальна)	(-1, 1)	$f(s) = \dfrac{e^{a \cdot s} - e^{-a \cdot s}}{e^{a \cdot s} + e^{-a \cdot s}}$
Лінійна з насиченням (крокова)	[-1, 1]	$f(s) = \begin{cases} -1, \text{ якщо } s \le -1 \\ s, \text{ якщо } -1 < s < 1 \\ 1, \text{ якщо } s \ge 1 \end{cases}$ $f(s) = \begin{cases} -1, \text{ якщо } s \le -1 \\ s, \text{ якщо } -1 < s < 1 \\ 1, \text{ якщо } s \ge 1 \end{cases}$
Напівлінійна з насиченням	[0, 1)	$f(s) = \begin{cases} 0, \text{ якщо } s \le 0 \\ s, \text{ якщо } 0 < s < 1 \\ 1, \text{ якщо } s \ge 1 \end{cases}$
Трикутна	[0, 1]	$f(s) = \begin{cases} 1 - \lvert s \rvert, \text{ якщо } \lvert s \rvert \le -1 \\ 0, \text{ якщо } \lvert s \rvert > 1 \end{cases}$
Радіальна базисна (Гаусова)	(0, 1]	$f(s) = e^{s^2}$

Нейронна мережа являє собою сукупність окремих нейронів, взаємопов'язаних між собою деяким фіксованим чином. При цьому взаємозв'язок нейронів визначається чи задається структурою *(топологією)* нейронної мережі. З точки зору топології нейронні мережі можуть бути повнозв'язними, багатошаровими і слабозв'язними. Структуру нейромережі можна описати таким чином (рис. 9.2). Нейромережа складається з декількох шарів: *вхідний*, *внутрішні (приховані)* і *вихідний шари*. Вхідний шар реалізує зв'язок з вхідними даними, вихідний – з вихідними. Внутрішніх шарів може бути від одного і більше. У кожному шарі міститься декілька нейронів.

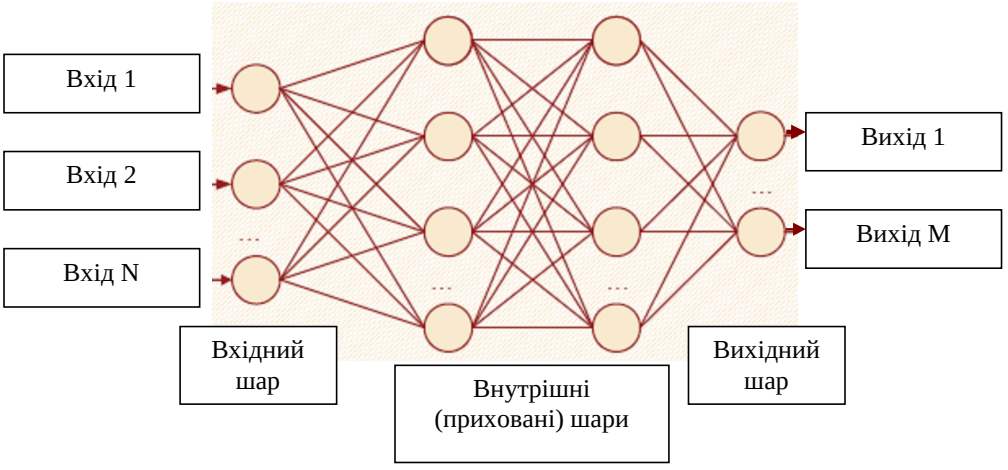

Рис. 9.2. Структура нейромережі

На сьогодні існує велике число конфігурацій нейронних мереж з різними принципами функціонування, які орієнтовані на вирішення певних завдань. До найбільш поширених нейронних мереж належать:

- *Перцептрон Розенблата*
- *Багатошаровий перцептрон*
- *Мережа Джордана*
- *Мережа Елмана*
- *Мережа Хемінга*
- *Гібридна*

У свою чергу, багатошарові нейронні мережі можуть бути таких типів:

- *Монотонні* – кожен шар (крім вихідного) додатково розбивається на 2 блоки: збуджуючий і гальмуючий. Аналогічно

розбиваються і зв’язки між блоками: на збуджуючі і гальмуючі. При цьому як функції активації можуть бути використані тільки монотонні функції (табл. 9.1).

- *Нейронні мережі зі зворотними зв’язками* – інформація з наступних шарів може передаватися на нейрони попередніх шарів.
- *Нейронні мережі без зворотних зв’язків* – інформація з наступних шарів не може передаватися на нейрони попередніх шарів.

2. Процес побудови і використання нейромережевих моделей

Процес побудови і використання нейромережевих моделей складається з таких етапів:

1) вибір типу і структури нейронної мережі для розв’язку поставленої проблеми (синтез структури нейронної мережі);

2) навчання нейронної мережі (визначення чисельних значень ваг кожного з нейронів) на основі наявної інформації про розв’язок даної задачі експертом чи даних про розв’язок задачі в минулому;

3) перевірка нейронної мережі на основі використання деякого контрольного прикладу (необов’язковий етап);

4) використання навченої нейронної мережі для розв’язку поставленої проблеми.

Вибір виду і структури нейронної мережі визначається специфікою розв’язуваної задачі. При цьому для розв’язку окремих типів практичних задач розроблені оптимальні конфігурації нейронних мереж, що найбільше адекватно відбивають особливості відповідної проблемної області.

Призначення і підготовка навчальної вибірки

Нейромережа здатна імітувати певний процес. Будь-яка зміна входів нейромережі веде до зміни її виходів. Причому виходи нейромережі однозначно залежать від її входів.

Перш ніж використовувати нейромережу, її необхідно навчити. Задача навчання тут рівносильна задачі апроксимації функції, тобто відновлення функції за окремо взятими її точками – таблично заданої функції. Таким чином, для навчання потрібно

підготувати таблицю з вхідними значеннями і відповідними їм вихідними значеннями, тобто підготувати навчальну вибірку. За такою таблицею нейромережа сама знаходить залежності вихідних полів від вхідних. Далі ці залежності можна використовувати, подаючи на вхід нейромережі деякі значення. На виході будуть відновлені залежні від них значення.

Схема навчання наведена на рис. 9.3.

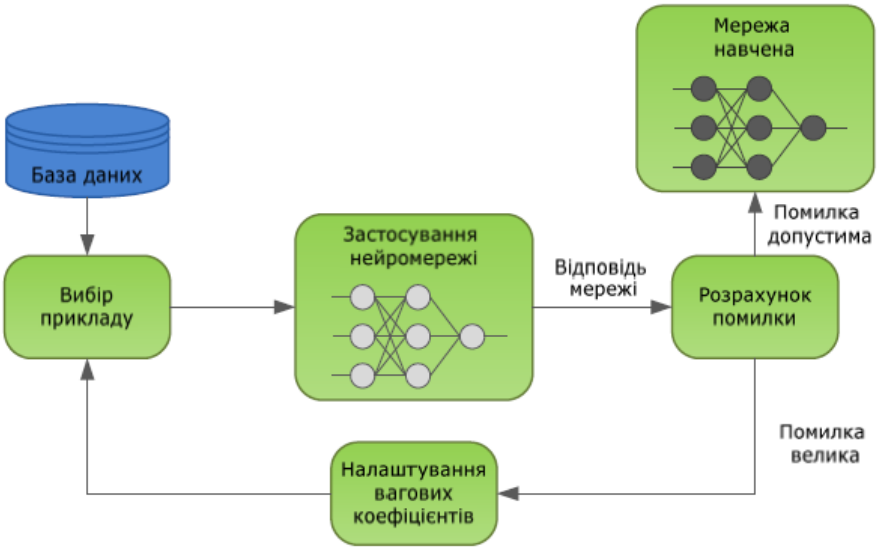

Рис. 9.3. Схема навчання нейронної мережі

Важливим є наступне. Навчальна вибірка не повинна містити суперечностей, оскільки нейромережа однозначно зіставляє вихідні значення вхідним. Після навчання на вхід нейромережі необхідно подавати значення з діапазону, на якому вона навчалася. Наприклад, якщо при навчанні нейромережі на один з її входів подавалися значення від 0 до 100, то надалі слід на цей вхід подавати значення з діапазону від 0 до 100. Допускається подавати значення, що лежать поряд з діапазоном.

Налаштування призначення полів

На самому початку роботи з нейромережею потрібно визначитися, що є її входами, а що виходами. Передбачається, що у нас вже є таблиця з навчальною вибіркою. Зазвичай для її підготовки користуються методами очищення і трансформації даних – *редагуються аномалії, заповнюються або віддаляються пропуски, усуваються дублікати і суперечності, проводиться квантування*

і таблична заміна, дані приводяться до ковзаючого вікна, перетворюється формат даних.

Нормалізація значень полів

Після того, як вказані вхідні і вихідні поля, слідує нормалізація даних у навчальній вибірці. Метою нормалізації значень полів є перетворення даних до вигляду, найбільш відповідному для обробки алгоритмом. Для таких обробників, як нейронна мережа, дерево рішень, лінійна модель прогнозування, дані, що поступають на вхід, повинні мати числовий тип, а їх значення повинні бути розподілені в певному діапазоні.

Нормалізатор може перетворити дискретні дані до набору унікальних індексів або значень, що лежать у довільному діапазоні, до діапазону [0..1]. Для нейромережі доступні такі види нормалізації полів.

1. *Лінійна нормалізація.* Використовується тільки *для неперервних числових полів*. Дозволяє привести числа до діапазону [min..max], тобто мінімальному числу з початкового діапазону відповідатиме min, а максимальному – max. Інші значення розподіляються між min і max.

2. *Унікальні значення.* Використовується *для дискретних значень*. Такими є рядки, числа або дати, задані дискретно. Щоб перевести неперервні числа в дискретні, можна, наприклад, скористатися обробкою «квантування». Так слід поступати для тих величин, *для яких можна задати відношення порядку*, тобто, якщо для двох будь-яких дискретних значень можна вказати, яке більше, а яке менше. Тоді всі значення необхідно розташувати в порядку зростання. Далі вони нумеруються по порядку, і значення *замінюються їх порядковим номером*.

3. *Бітова маска.* Використовується *для дискретних значень*. Цей вид нормалізації слід використовувати для величин, які можна тільки порівнювати на рівність або нерівність, але не можна сказати, яке більше, а яке менше. Усі значення замінюються порядковими номерами, а номер розглядається в двійковому вигляді або у вигляді маски з нулів і одиниць. Тоді кожна позиція маски розглядається як окреме поле, що містить нуль або одиницю. До такого поля можна застосувати лінійну нормалізацію, тобто

замінити нуль деяким мінімальним значенням, а одиницю – максимальним. Після такої нормалізації на вхід нейромережі подаватиметься не одне це поле, а стільки полів, скільки розрядів у масці.

Приклади нормалізації полів

1. *Лінійна нормалізація*. Приведемо значення до діапазону [0..1].

Поле до нормалізації	Поле після нормалізації
–5	0
2,3	0,81111
1,1	0,67778
4	1
3,5	0,94444

Графічно таке перетворення можна представити так:

2.*Унікальні значення*

Поле до нормалізації	Поле після нормалізації
Маленький	1
Середній	2
Великий	3
Величезний	4

3. *Бітова маска*. Замінимо значення бітовою маскою і приведемо до діапазону [–1..1].

Поле до нормалізації	Маска	Поля після нормалізації	
Київ	00	–1	–1
Чернігів	10	1	–1
Донецьк	01	–1	1
Суми	11	1	1

Крім того, існує Налаштування нормалізації полів за замовчуванням, тобто етап нормалізації полів можна пропустити. У

цьому випадку нормалізація буде проведена автоматично залежно від типу полів:

– для дискретного – нормалізація бітовою маскою із способом кодування – комбінація бітів.

– неперервного – лінійна нормалізація в діапазоні [–1..1].

Налаштування навчальної вибірки

Після нормалізації полів слід побудувати навчальну вибірку. Навчальну вибірку розбивають на дві множини – *навчальну* і *тестову*. Навчальна множина включає записи (приклади), які використовуватимуться безпосередньо для навчання мережі, тобто міститимуть вхідні і бажані вихідні *(цільові*) значення. Тестова множина також включає записи (приклади), вхідні і бажані, що містять вихідні (цільові) значення, але що використовується не для навчання мережі, а для перевірки результатів навчання. Розбивати початкову вибірку на ці множини можна двома способами: або по порядку, або випадково. Якщо розбиття відбувається по порядку, то тестова множина вибирається або з початку, або з кінця початкової вибірки.

Налаштування структури нейромережі

Далі задаються параметри, що визначають структуру нейронної мережі – *кількість прихованих шарів і нейронів у них,* а також *активаційна функція нейронів.*

До вибору кількості прихованих шарів і кількості нейронів для кожного прихованого шару потрібно *підходити обережно*. Хоча дотепер не вироблені чіткі критерії вибору, дати деякі загальні рекомендації все ж таки можливо. Вважається, що задачу будь-якої складності можна вирішити за допомогою *двошарової нейромережі*, тому конфігурація з кількістю прихованих шарів, перевищуючих 2, навряд чи виправдана. Для вирішення багатьох задач цілком підійде *одношарова нейронна мережа*. При виборі кількості нейронів слід керуватися таким правилом: «*кількість зв'язків між нейронами повинна бути значно менша кількості прикладів у навчальній множині».* Кількість зв'язків розраховується як зв'язок кожного нейрона *зі всіма нейронами сусідніх шарів, включаючи зв'язки на вхідному і вихідному шарах.*

Досить велика кількість нейронів може привести до так званого «перенавчання» мережі, коли вона видає гарні результати на прикладах, що входять у навчальну вибірку, але практично не працює на інших прикладах.

Алгоритми навчання нейромережі

Після настройки конфігурації мережі слід вибрати **алгоритм її навчання**.

Метод зворотного розповсюдження помилки – ітеративний градієнтний алгоритм навчання, який використовується з метою *мінімізації середньоквадратичного відхилення поточних значень виходів мережі від тих, що вимагаються*. Однією з найважливіших властивостей алгоритму зворотного розповсюдження помилки є *висока стійкість*, а отже, *надійність*. Хоча нейронні мережі, які використовують алгоритм зворотного розповсюдження, будучи могутнім інструментом пошуку закономірностей, прогнозування і якісного аналізу, одержали широке розповсюдження, їм властиві деякі *недоліки*. До них належить *невисока швидкість збіжності* (велике число необхідних ітерацій), що робить *процес навчання дуже довгим*, і тому даний алгоритм виявляється *непридатним для широкого кола задач, вимагаючих швидкого рішення*. Інші алгоритми навчання нейромереж хоча і працюють швидше, у більшості випадків володіють меншою стійкістю.

Для алгоритму зворотного розповсюдження помилки *потрібно вказати два параметри*:

– *швидкість навчання* – визначає величину кроку при ітераційній корекції ваг у нейронній мережі (рекомендується в інтервалі 0…1). При великій величині кроку збіжність буде швидшою, але є небезпека «перестрибнути» через рішення. З іншої сторони, при малій величині кроку навчання зажадає дуже багато ітерацій. На практиці величина кроку береться пропорційній крутизні схилу так, що алгоритм сповільнюється поблизу мінімуму. Правильний вибір швидкості навчання залежить від конкретної задачі і звичайно робиться досвідченим шляхом;

– *момент* – задається в інтервалі 0…1. Рекомендується значення 0,9 ±0,1;

– *метод Resilent Propogation (Rprop) – еластичне розповсюдження*. Алгоритм використовує так зване *«навчання за епохами»*, коли корекція ваг відбувається після пред'явлення мережі всіх прикладів з навчальної вибірки. Перевага цього методу полягає в тому, що він забезпечує збіжність, а отже, і *навчання мережі в 4–5 разів швидше, ніж алгоритм зворотного розповсюдження*.

Для алгоритму Resilient Propagation вказуються параметри:

– *крок спуску* – коефіцієнт збільшення швидкості навчання, який *визначає крок збільшення швидкості навчання* при недосягненні алгоритмом оптимального результату;

– *крок підйому* – коефіцієнт *зменшення швидкості навчання*. Задається крок зменшення швидкості навчання у разі пропуску алгоритмом оптимального результату.

Далі необхідно задати умови, при виконанні яких навчання буде припинено:

– *вважати приклад розпізнаним, якщо помилка менша* – якщо розузгодження між еталонним і реальним виходом мережі стає менше заданого значення, то приклад вважається правильно розпізнаним;

– *після досягнення епохи* – установка даного режиму дозволяє задати число епох (циклів навчання), після досягнення якого навчання зупиняється незалежно від величини помилки. Якщо прапорець скинутий, то навчання продовжуватиметься, доки помилка не стане менша заданого значення, але при цьому є вірогідність зациклення, коли помилка ніколи не буде досягнута. Тому бажано встановити прапорець по «Досягненню епохи», щоб алгоритм був зупинений у будь-якому випадку;

– *навчальна множина* – зупинка навчання проводиться після досягнення на навчальній множині заданої середньої помилки, максимальної помилки або відсотка розпізнаних прикладів. Розпізнаним вважається приклад, для якого відхилення розрахункового і реального значення не більше параметра «Вважати приклад розпізнаним, якщо помилка менша»;

– *тестова множина* – зупинка навчання проводиться після досягнення на тестовій множині заданої середньої помилки, максимальної помилки або відсотка розпізнаних прикладів. Розпізнаним вважається приклад, для якого відхилення розрахункового і реального значення не більше параметра «Вважати приклад розпізнаним, якщо помилка менша».

Зупинка навчання відбувається після досягнення будь-якої із заданих умов зупинки.

Тепер усе готово до процесу навчання мережі. На початку вся вага нейромережі ініціалізується випадковими значеннями. Після навчання ця вага приймає певні значення. Навчання можна вважати з великою імовірністю успішним, якщо відсоток розпізнаних прикладів на навчальній і тестовій множинах достатньо великий (близький до 100 %).

Далі здійснюється перевірка нейронної мережі на основі використання деякого контрольного прикладу (може бути необов'язковий етап). При позитивній верифікації навчена нейронна мережа використовується для розв'язання поставленої проблеми.

Карти Кохонена

Карта, що самоорганізується, карта Кохонена – це змагальна нейронна мережа з навчанням без вчителя. У процесі навчання карт Кохонена на виходи також подаються дані та індикатори, але *при цьому мережа підстроюється під закономірності у вхідних даних*, а не під еталонне значення виходу. Навчання при цьому полягає в підстроюванні внутрішніх параметрів мережі для більшого збігу з вхідними даними. Після навчання така мережа візуально відображає багатомірні вхідні дані на площині нейронів. Маючи таке представлення даних, можна наглядно побачити наявність або відсутність взаємозв'язку у вхідних даних.

Карти Кохонена можуть використовуватися для вирішення таких завдань, як моделювання, прогнозування, пошук закономірностей у великих масивах даних, виявлення наборів незалежних ознак і стискування інформації.

Алгоритм функціонування карт, що самоорганізуються, (Self Organizing Maps – SOM), є одним з варіантів *кластеризації*

багатовимірних векторів – алгоритм проектування зі збереженням топологічної подібності.

Прикладом таких алгоритмів може служити алгоритм k – найближчих середніх (k – means). Важливою відмінністю алгоритму SOM є те, що в нім усі нейрони (вузли, центри класів) впорядковані в деяку структуру (зазвичай двовимірну сітку). При цьому в ході навчання модифікується не лише нейрон-переможець (нейрон карти, який найбільшою мірою відповідає вектору входів і визначає, до якого класу відноситься приклад), але і його сусіди, хоча і меншою мірою. За рахунок цього SOM можна вважати одним з методів проектування багатовимірного простору в простір з нижчою розмірністю. При використанні цього алгоритму вектори, близькі в початковому просторі, виявляються поруч і на отриманій карті.

SOM має на увазі використання впорядкованої структури нейронів. Зазвичай використовуються одно- і двовимірні сітки. При цьому кожен нейрон є n-мірний вектор-стовпець, де n визначається розмірністю початкового простору (розмірністю вхідних векторів). Застосування одно- і двовимірних сіток пов’язано з тим, що виникають проблеми при відображенні просторових структур більшої розмірності (при цьому знову виникають проблеми з пониженням розмірності до двовимірної, уявної на моніторі).

Зазвичай нейрони розташовуються у вузлах двовимірної сітки з прямокутними або шестикутними осередками. При цьому, як було сказано вище, нейрони також взаємодіють один з одним. Величина цієї взаємодії визначається відстанню між нейронами на карті.

При реалізації алгоритму SOM заздалегідь задається конфігурація сітки (прямокутна або шестикутна), а також кількість нейронів у мережі. Деякі джерела рекомендують використати максимально можливу кількість нейронів у карті. При цьому початковий радіус навчання (neighbourhood в англомовній літературі) значною мірою впливає на здатність узагальнення за допомогою отриманої карти. У разі коли кількість вузлів карти перевищує кількість прикладів у навчальній вибірці, успіх використання алгоритму великою

мірою залежить від відповідного вибору початкового радіусу навчання. Проте у разі коли розмір карти складає десятки тисяч нейронів, час, потрібний на навчання карти, зазвичай буває занадто великим для вирішення практичних завдань. Таким чином, необхідно досягати допустимий компроміс при виборі кількості вузлів.

Відмінність у підготовці навчальної вибірки в цьому алгоритмі полягає в тому, що вихідні поля в такій вибірці можуть бути відсутніми зовсім. Навіть якщо в навчальній вибірці будуть присутніми вихідні поля, вони не братимуть участь при навчанні нейроятера, проте вони братимуть участь при відображенні карт. Нормалізація полів тут така ж, як звичайно для нейромереж. Навчальна вибірка настроюється також, як і для нейромережі і дерева рішень. Перед початком навчання карти потрібно проініціалізувати вагові коефіцієнти нейронів. Вдало вибраний спосіб ініціалізації може істотно прискорити навчання і привести до отримання якісніших результатів. Існують три способи ініціації початкових ваг.

– *випадковими значеннями*, коли усім вагам даються малі випадкові величини;

– *з навчальної множини*, коли початковими значеннями задаються значення випадково вибраних прикладів з навчальної вибірки;

– *з власних векторів*. У цьому випадку ваги ініціюються значеннями векторів, лінійно впорядкованих уздовж лінійного підпростору, що проходить між двома головними власними векторами початкового набору даних.

Отриману в результаті навчання карту можна представити у вигляді пирога, кожен шар якого є розфарбовуванням, породженим однією з компонент початкових даних. Отриманий набір розфарбовувань може використовуватися для аналізу закономірностей, наявних між компонентами набору даних. Після формування карти виходить набір вузлів, який можна відобразити у вигляді двовимірної картинки. При цьому кожному вузлу карти можна поставити у відповідність ділянку на малюнку (чотирьох або шестикутну), координати якої визначаються координатами

відповідного вузла решітки. Тепер для візуалізації залишається тільки визначити колір комірок цієї картинки. Для цього і використовуються значення компонент. Найпростіший варіант – використання градацій сірого. У цьому випадку комірки, що відповідають вузлам карти, в які потрапили елементи з мінімальними значеннями компонента або не потрапило взагалі жодного запису, будуть зображені чорним кольором, а комірки, в які потрапили записи з максимальними значеннями такого компонента, відповідатимуть осередку білого кольору. Зручнішим є використання для розфарбовування кольорової палітри. У принципі можна використати будь-яку градієнтну палітру для розфарбовування. Отримані розфарбовування в сукупності утворюють атлас, що відображає розташування компонент, зв'язки між ними, а також відносне розташування різних значень компонент.

3. Гібридна модель нейронної мережі та її реалізація в пакеті Fuzzy Logic Toolbox системи MATLAB

Гібридна мережа являє собою багатошарову нейронну мережу спеціальної структури без зворотних зв'язків, у якій використовуються звичайні (не нечіткі) сигнали, ваги і функції активації, а виконання операції підсумовування (6.1) засновано на використанні фіксованої Т-норми, Т-конорми чи деякої іншої неперервної операції. При цьому значення входів, виходів і ваг гібридної нейронної мережі являють собою дійсні числа з відрізка [0, 1].

Основна ідея, покладена в основу моделі гібридних мереж, полягає в тому, щоб використовувати існуючу вибірку даних для визначення параметрів функцій приналежності, що найкраще відповідають деякій системі нечіткого виводу. При цьому для знаходження параметрів функцій приналежності використовуються відомі процедури навчання нейронних мереж.

У пакеті Fuzzy Logic Toolbox системи MATLAB модель гібридної мережі реалізована як адаптивна система нейро-нечіткого виводу ANFIS.

З одного боку, гібридна мережа ANFIS являє собою нейронну мережу з єдиним виходом і декількома входами, що являють

собою нечіткі лінгвістичні змінні. При цьому терми вхідних лінгвістичних змінних описуються стандартними для системи MATLAB функціями приналежності, а терми вихідної змінної представляються лінійною чи постійною функцією приналежності.

З іншого – гібридна мережа ANFIS являє собою систему нечіткого виводу FIS типу Сугено нульового чи першого порядку, у якій кожне з правил нечітких продукцій має постійну вагу, рівну 1. У системі MATLAB користувач має можливість редагувати і налаштовувати гібридні мережі ANFIS аналогічно системам нечіткого виводу, використовуючи всі розглянуті раніше засоби пакета Fuzzy Logic Toolbox.

У пакеті Fuzzy Logic Toolbox системи MATLAB гібридні мережі створюються у формі адаптивних систем нейронечіткого виводу. При цьому розробка і дослідження гібридних мереж виявляються можливою в:

- інтерактивному режимі за допомогою спеціального графічного редактора адаптивних мереж, що одержав назву редактора ANFIS;
- режимі командного рядка за допомогою введення імен відповідних функцій з необхідними аргументами безпосередньо у вікно команд системи MATLAB. Для роботи в режимі командного рядка призначені спеціальні функції.

Редактор ANFIS дозволяє створювати чи завантажувати конкретну модель адаптивної системи нейронечіткого виводу, виконувати її навчання, візуалізувати її структуру, змінювати і налагоджувати її параметри, а також використовувати налагоджену мережу для одержання результатів нечіткого виводу.

ANFIS-редактор

Завантаження ANFIS-редактора здійснюється за командою *anfisedit.* У результаті виконання цієї команди з’явиться графічне вікно, зображене на рис. 6.4. На цьому ж рисунку зазначено функціональні області ANFIS-редактора, опис яких приведено на рис. 9.4.

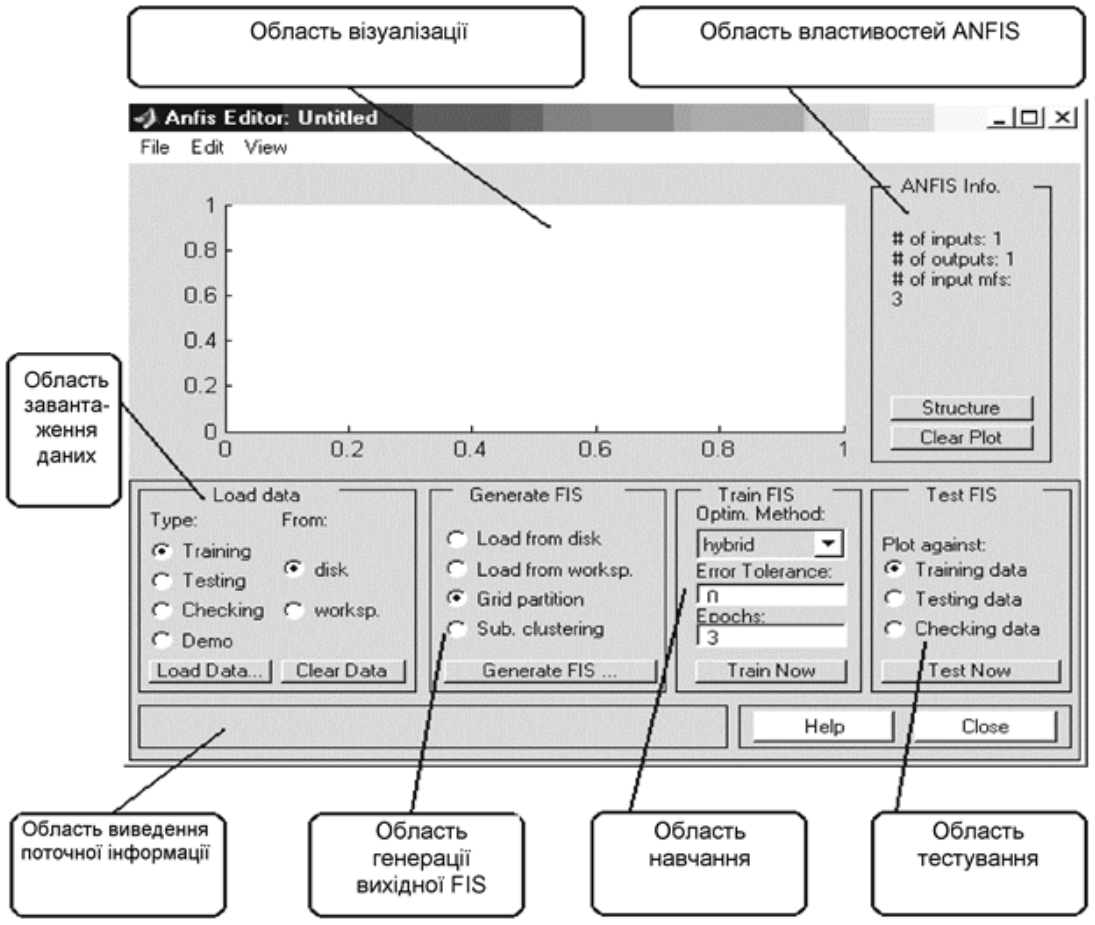

Рис. 9.4. Графічне вікно ANFIS-редактора

ANFIS-редактор містить 3 верхніх меню – File, Edit і View, область візуалізації, область властивостей ANFIS, область завантаження даних, область генерування вихідної системи нечіткого логічного виведення, область навчання, область тестування, область виведення поточної інформації, а також кнопки Help і Close, що дозволяють викликати вікно довідки і закрити ANFIS-редактор, відповідно.

Меню та команди Anfis редактора

Меню Edit:

Команда **Undo** скасовує раніше зроблену дію. Виконується також по натисканню Ctrl+Z.

Команда **FIS Properties**… відкриває FIS-редактор. Ця команда може бути також виконана натисканням Ctrl+1.

Команда **Membership Functions**… відкриває редактор функцій приналежності. Ця команда може бути також виконана натисканням Ctrl+2.

Команда **Rules…** відкриває редактор бази знань. Ця команда може бути також виконана натисканням Ctrl+3.

Команда **Anfis…** відкриває ANFIS-редактор. Ця команда може бути також виконана натисканням Ctrl+3. Зазначимо, що ця команда, запущена з ANFIS-редактора, не приводить до виконання будь-яких дій, бо цей редактор уже відкритий. Однак у меню Editінших GUI-модулів, використовуваних із системами нечіткого логічного виведення, додається команда Anfis…, що дозволяє відкрити ANFIS-редактор із цих модулів.

Область візуалізації містить два типи інформації: при навчанні системи – крива навчання у виді графіка залежності помилки навчання від порядкового номера ітерації; при завантаженні даних і тестуванні системи – експериментальні дані і результати моделювання.

Експериментальні дані і результати моделювання виводяться у виді множини точок у двовимірному просторі. При цьому по вісі абцис відкладається порядковий номер рядка даних у вибірці (навчальній, тестовій або контрольній), а по осі ординат – значення вихідної змінної для даного рядка вибірки. Використовуються такі маркери: блакитна точка (.) – тестова вибірка; блакитна окружність (o) – навчальна вибірка; блакитний плюс (+) – контрольна вибірка; червона зірочка (*) – результати моделювання.

В області властивостей ANFIS (ANFIS info) виводиться інформація про кількість вхідних і вихідних змінних, про кількість функцій приналежності для кожної вхідної змінної, а також про кількість рядків у вибірках. У цій області розташовані дві кнопки Structurei Clear Plot.

Натискання кнопки Structure відчиняє нове графічне вікно, у якому система нечіткого логічного виведення представлена у виді нейро-нечіткої мережі. Натискання кнопки Clear Plot дозволяє очистити область візуалізації.

В області завантаження даних (Load data) розташовані: меню вибору типу даних (Type), що містить альтернативи (Traning – навчальна вибірка; Testing – тестова вибірка; Checking – контрольна вибірка; Demo – демонстраційний приклад); меню вибору джерела даних (From), що містить альтернативи (disk – диск;

worksp – робоча область MatLab); кнопка завантаження даних Load Data…, по натисканню якої з'являється діалогове вікно вибору файлу, якщо завантаження даних відбувається з диска, або вікно введення ідентифікатора вибірки, якщо завантаження даних походить з робочої області; кнопка очищення даних Clear Data.

В області генерування (Generate FIS) розташовані меню вибору способу створення вихідної системи нечіткого логічного виведення. Меню містить наступні альтернативи: Load from disk – завантаження системи з диска; Load from worksp. – завантаження системи з робочої області MatLab; Grid partition – генерування системи по методу ґрат (без кластеризації); Sub. clustering – генерування системи за методом субкластеризації.

В області також розташована кнопка Generate, по натисканню якої генерується вихідна система нечіткого логічного виведення.

При виборі **Load from disk** з'являється стандартне діалогове вікно відкриття файлу.

При виборі Load from worksp. з'являється стандартне діалогове вікно введення ідентифікатора системи нечіткого логічного виведення.

При виборі **Grid partition** з'являється вікно введення параметрів методу ґрат, у якому потрібно вказати кількість термів для кожен вхідний змінної і тип функцій приналежності для вхідних і вихідної змінних.

При виборі **Sub clustering** з'являється вікно введення наступних параметрів методу субкластеризації: Range of influence – рівні впливу вхідних змінних; Squash factor – коефіцієнт пригнічення; Accept ratio – коефіцієнт, що встановлює, у скільки разів потенціал даної точки повинен бути вище потенціалу центра першого кластера для того, щоб центром одного з кластерів була призначена розглянута точка; Reject ratio – коефіцієнт, що встановлює, у скільки разів потенціал даної точки повинен бути нижче потенціалу центра першого кластера, щоб розглянута точка була виключена з можливих центрів кластерів.

В області навчання (Train FIS) розташовані меню вибору методу оптимізації (Optim. method), поле завдання необхідної точності навчання (Error tolerance), поле завдання кількості ітерацій навчання (Epochs) і кнопка Train Now, натискання якого запускає режим навчання. Проміжні результати навчання виводяться в область візуалізації й у робочу область MatLab. У ANFIS-редакторі реалізовані два методи навчання: backpropa – метод зворотного поширення помилки, заснований на ідеях методу найшвидшого спуска; hybrid – гібридний метод, що поєднує метод зворотного поширення помилки з методом найменших квадратів.

В області тестування (Test FIS) розташовані меню вибору вибірки і кнопка Test Now, при натисканні на яку відбувається тестування нечіткої системи з виведення результатів в область візуалізації.

Область виведення поточної інформації: у цій області виводиться найбільш істотна поточна інформація, наприклад, повідомлення про закінчення виконання операцій, значення помилки чи навчання тестування тощо.

Питання для самоконтролю

1. Яке визначення нейронної мережі?
2. Що є концептуальною основою штучних нейронних мереж?
3. Які види функції активації нейрона існують?
4. Що називають топологією нейронної мережі?
5. Які різновиди нейронних мереж є найбільш поширеними?
6. Які основні етапи побудови і застосування нейронних мереж?
7. Які етапи навчання мережі можна виокремити?
8. Які є алгоритми навчання нейронної мережі?
9. Яка основна ідея покладена в основу гібридної моделі нейронної мережі?
10. Що являє собою адаптивна система нейронечіткого виводу ANFIS?

ЛАБОРАТОРНА РОБОТА 9.1

Тема. Розробка інтелектуальних систем на основі моделей нейроних мереж.

Мета роботи: освоїти методику проектування і побудови інтелектуальних систем на основі моделей нечітких нейронних мереж.

Методичні рекомендації

Для ілюстрації процесу розробки гібридної мережі в системі MATLAB розглянемо задачу побудови адаптивної системи нейронечіткого виводу для апроксимації деякої виробничої функції типу Кобба-Дугласа:

$$q = F(x_1, x_2) = 2\sqrt{x_1}\sqrt[4]{x_2},$$

яка описує залежність між витратами двох виробничих факторів (наприклад, грошей і праці) у кількості x_1 і x_2 та кількістю виготовленої продукції q у наперед визначених одиницях.

Загальна послідовність процесу розробки моделі гібридної мережі може бути представлена в наступному вигляді.

Крок 1. Для початку за допомогою редактора-відладчика m-файлів або будь-якого текстового редактора підготуємо навчальні дані, що містять 6-ти рядків з трьох значень – два вхідних і одне вихідне значення, наприклад, наступного виду:

```
1  2      2.37
4  1      4.00
1  4      2.82
5  5      6.68
2  4      4.00
4  3      5.26
3  1      3.46
4  4      5.65
4  2      4.80
```

Числа в рядку розділяються клавішею «**Space**», а перехід на наступний рядок відбувається натисненням клавіші «**Enter**». Слід

зберегти цей файл у папці work каталогу Matlab6, наприклад, з назвою **train.txt**.

Взагалі, вхідні дані представляють собою звичайну числову матрицю розмірності $m \times (n+1)$, у якій кількість рядків m відповідає обсягу вибірки, перші n стовпців – значенням вхідних змінних моделі, а останній стовпчик – значенню вихідної змінної. Відповідно до правил системи MATLAB окремі значення матриці відокремлюються пробілами, а кожен рядок матриці завершується символом *«перевід каретки»* (клавіша <Enter>).

Хоча по кількості рядків матриці вхідних даних не існує формальних рекомендацій, прийнято вважати, що якість навчання гібридної мережі, а отже, і точність одержуваних результатів пропорційно залежить від обсягу навчальної вибірки. Що стосується кількості стовпців матриці вхідних даних, то слід зазначити можливі проблеми з працездатністю системи MATLAB, якщо кількість вхідних змінних перевищує 5–6.

Початкові дані, що завантажуються, можуть бути одним з таких типів:

– навчальні дані **(Training)** – обов’язкові дані, що використовуються для побудови гібридної мережі;

– тестові дані **(Testing)** – необов’язкові дані, що використовуються для тестування побудованої гібридної мережі з метою перевірки якості функціонування побудованої гібридної мережі;

– перевірочні дані **(Checking)** – необов’язкові дані, що використовуються для перевірки побудованої гібридної мережі з метою з’ясування факту перенавчання мережі;

– демонстраційні дані **(Demo)** – дозволяють завантажити один з демонстраційних прикладів гібридної мережі.

Тому створимо ще і файл тестових даних **test.txt**. За аналогією зі створенням файлу train.txt введемо в нього такі дані:

```
2  1      2.82
3  2      4.12
1  3      2.63
2  3      3.72
3  3      4.56
4  2      4.75
3  4      4.90
```

Крок 2. Головне меню редактора ANFIS достатньо просте і призначене для роботи з попередньо створеною системою нечіткого виводу. Основну частину графічного інтерфейсу займає вікно візуалізації даних, що розташоване нижче головного меню (рис. 9.1.1). Для знову створюваної гібридної мережі це вікно не містить ніяких даних. Для створення гібридної мережі необхідно завантажити дані. Для цієї мети слід скористатися кнопкою Load Data в лівій нижній частині графічного вікна. При цьому дані можуть бути завантажені із зовнішнього файлу (disk) чи з робочої області (worksp). У першому випадку необхідно попередньо створити файл із вхідними даними (файл train), що являє собою звичайний текстовий файл. Після завантаження файлу із навчальними даними в редактор ANFIS у робочому вікні редактора буде зображений графік, ордината якого відображає значення вхідної математичної функції (кількість продукції у прикладі).

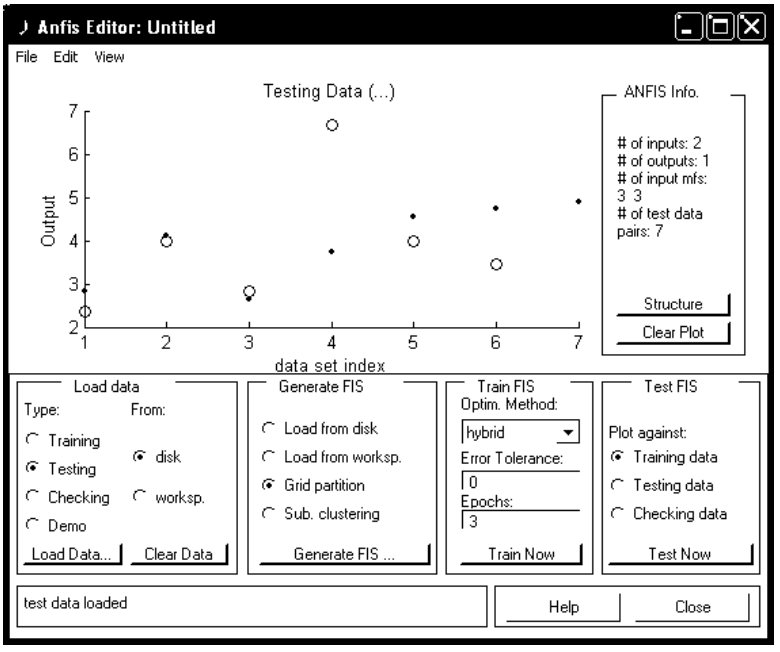

Рис. 9.1.1. Графічний інтерфейс редактора ANFIS після завантаження файлу function.dat з навчальними даними

Завантажимо ще і тестові дані для перевірки. Для цього в лівому нижньому кутку екрану оберемо тип **testing**, натиснемо кнопку **Load data** і завантажимо файл **test.txt**.

Крок 3. Приступити до генерації структури системи нечіткого виводу FIS**.** Ця структура аналогічна структурі системи нечіткого

виводу типу Сугено. Тобто має у своєму складі лінгвістичні вхідні змінні; вихідну змінну, терми якої представляються у вигляді числових констант або лінійних функцій від вхідних змінних; систему правил виводу.

Крім того, можна завантажити структуру уже створеної FIS з диска **(Load from disk),** або з робочої області **(Load from worksp).** При створенні структури нової FIS можна незалежно розбити всі вхідні змінні на області їх значень **(Grid partision)** чи скористатися процедурою субтрактивної кластеризації для попередньої розбивки значень вхідних змінних на кластери близьких значень **(Sub. clustering).**

Після натискання кнопки **Generate FIS** викликається діалогове вікно з вказівкою числа і типу функцій приналежності для окремих термів вхідних змінних і вихідної змінної (рис. 9.1.2). У цьому вікні можна вибрати тип функцій приналежності і кількість термів кожної вхідної змінної, а також тип вихідної змінної. Установивши параметри генерації, наприклад, як на рис. 9.1.2, одержимо структуру FIS.

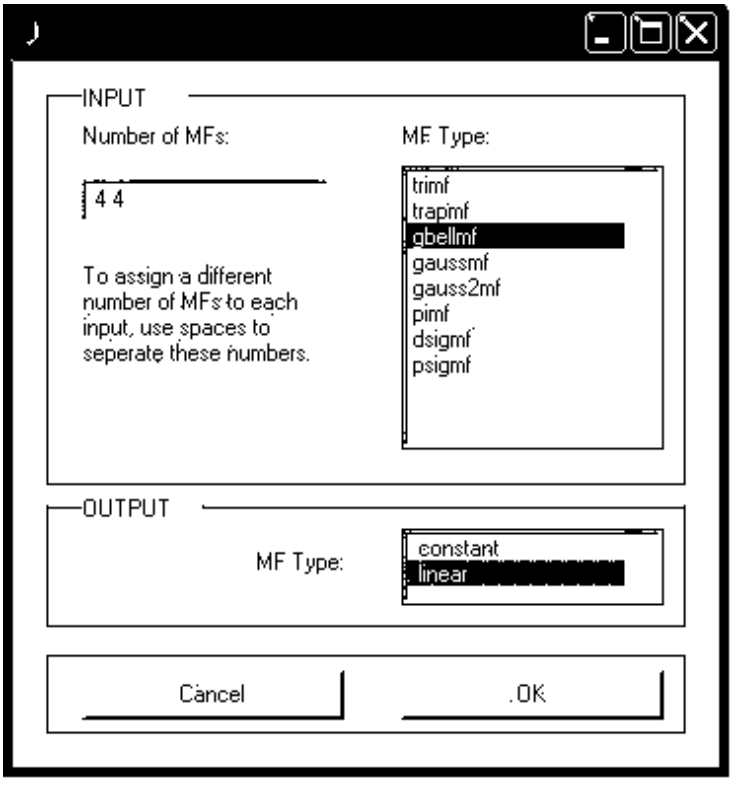

Рис. 9.1.2. Діалогове вікно для завдання кількості і типу функцій приналежності

Після генерації структури гібридної мережі можна її подивитись, натиснувши кнопку **Structure** в правій частині графічного вікна. Структура отриманої в результаті системи нечіткого виводу FIS відображається в окремому вікні і досить тривіальна за своїм видом. Проаналізуйте її зміст.

Крок 4. Перед навчанням гібридної мережі необхідно задати параметри навчання, для чого варто скористатися наступною групою опцій у правій нижній частині робочого вікна:

1. Вибрати метод навчання гібридної мережі – зворотного розповсюдження **(bacpropo)** чи гібридний **(hybrid)**, що представляє собою комбінацію методу найменших квадратів і методу спадання зворотного градієнта.

2. Установити рівень помилки навчання **(Error Tolerance)** – за замовчуванням значення 0 (змінювати не рекомендується).

3. Задати кількість циклів навчання (Epochs) – за замовчуванням значення 3 (рекомендується збільшити і для розглянутого прикладу задати його значення рівним 40).

6. Для навчання мережі варто натиснути кнопку **Trian Now**. При цьому хід процесу навчання ілюструється у вікні візуалізації у формі графіка – залежність помилки від кількості циклів навчання.

7. Аналогічно можуть бути виконані додаткові етапи тестування й перевірки гібридної мережі, для яких необхідно попередньо завантажити відповідні дані.

Крок 5. Подальше настроювання параметрів побудованої і навченої гібридної мережі може бути виконане за допомогою розглянутих раніше стандартних графічних засобів пакета Fuzzy Logic Toolbox. Для цього рекомендується зберегти створену систему нечіткого виводу в зовнішньому файлі з розширенням **fis**, після чого варто завантажити цей файл у редактор систем нечіткого виводу FIS. Можна працювати зі структурою системи нечіткого виводу безпосередньо в редакторі ANFIS.

При цьому також стають доступними редактор функцій приналежності системи нечіткого виводу (Membership Function Editor), редактор правил системи нечіткого виводу (Rule Editor), програма перегляду правил системи нечіткого виводу (Rule

Viewer) і програма перегляду поверхні системи нечіткого виводу (Surface Viewer). Проаналізуйте правила та функції приналежності, які були побудовані даною системою апроксимації виробничої функції. Спробуйте їх корегувати і слідкувати за зміною результатів.

Крок 6. Виконати аналіз точності побудованої нечіткої моделі гібридної мережі можна за допомогою перегляду поверхні відповідної системи нечіткого виводу. Для цього слід записати побудовану мережу у вигляді файлу з розширенням *.fis і завантажити його в системі нечіткого виводу fuzzyToolbox (рис. 9.1.3).

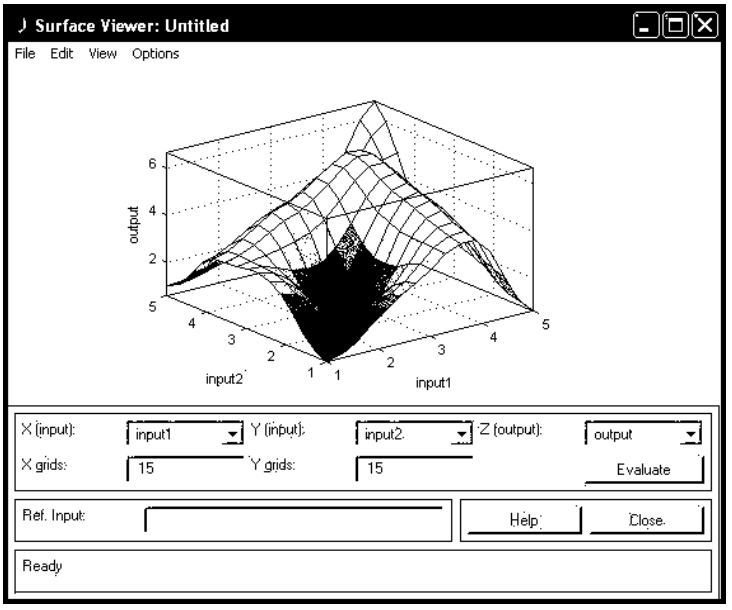

Рис. 9.1.3. Графічний інтерфейс перегляду поверхні згенерованої системи нечіткого виводу

Візуальний аналіз зображеного графіка з точним графіком функції $q = F(x_1, x_2) = 2\sqrt{x_1}\sqrt[4]{x_2}$ дозволяє судити про досить високий ступінь їх співпадання, що може свідчити про адекватність побудованої нечіткої моделі гібридної мережі.

Аналіз адекватності побудованої моделі можна виконати за допомогою перегляду правил відповідної системи нечіткого виводу (рис. 9.1.4).

Перевірка побудованої моделі гібридної мережі може бути виконана для декількох значень вихідної змінної. З цією метою необхідно ввести конкретне значення в поле вводу **Input** (наприк-

лад, значення 3 1), після натискання клавіші <Enter> за допомогою побудованої моделі буде отримане відповідне значення вихідної змінної (у даному випадку значення 3.46). Порівнюючи отримане значення з точним значенням функції 3.46, одержимо відносну помилку порядку 0 %.

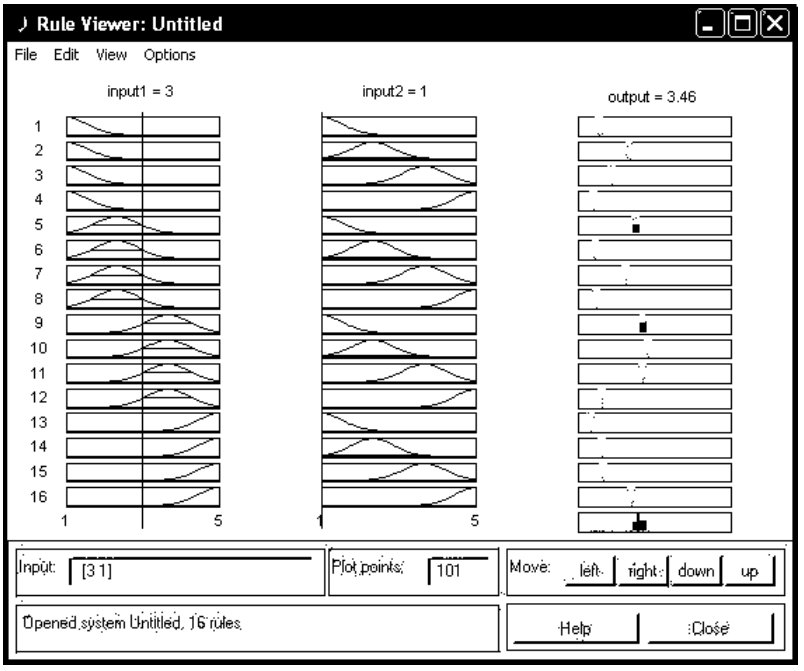

Рис. 9.1.4. Графічний інтерфейс перегляду правил згенерованої системи нечіткого виводу

Менш вдалою виявляється перевірка для значення вхідних змінних 4 3, для якого побудована модель пропонує значення 5.45 (табличне значення 5.26). Очевидно, цей факт свідчить не на користь адекватності побудованої нечіткої моделі і вимагає її додаткового настроювання.

У загальному випадку додаткове настроювання моделі може бути виконане декількома можливими способами. Найбільш прийнятними з них представляються наступні.

1. Підготовка і завантаження більшого за обсягом вибірки файлу з навчальними вхідними даними.

2. Підготовка і завантаження додаткового файлу з перевірочними вхідними даними, сформованими для пар значень розглянутої математичної функції, відсутніх у вибірці навчальних даних.

3. Редагування типів і значень параметрів функцій приналежності термів вхідної і вихідної змінних за допомогою редактора функцій приналежності системи MATLAB.

Проілюструємо третій спосіб додаткового настроювання побудованої нечіткої моделі гібридної мережі. На перший погляд він представляється найбільш природним з погляду можливості візуального контролю виконуваних змін параметрів. З цією метою відкриємо редактор функцій приналежності і методом підбору змінимо кількісні значення параметрів другої і третьої функції приналежності вхідної змінної, оскільки саме вони «працюють» при одержанні некоректного значення вихідної змінної: 5.45 для значення вхідних змінних 4 і 3 (рис. 9.1.5).

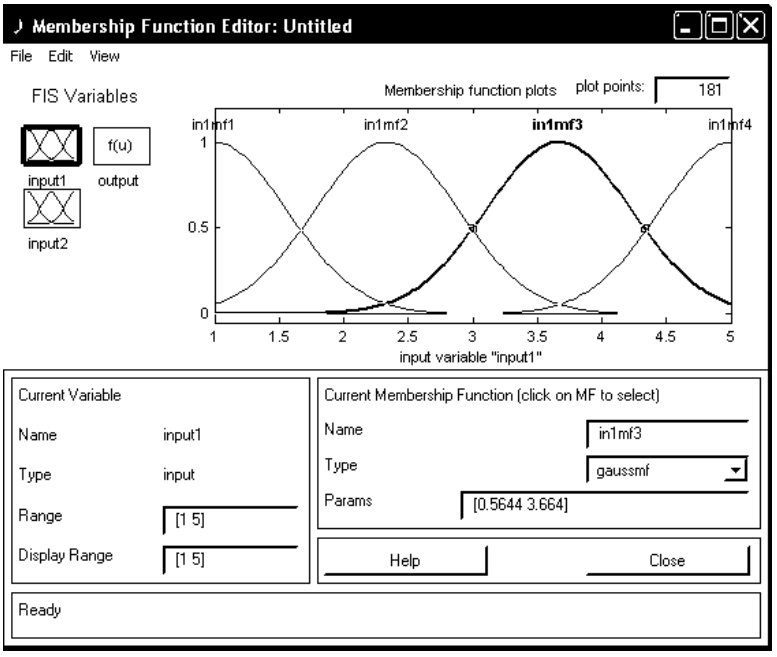

Рис. 9.1.5. Графічний інтерфейс редактора функцій приналежності побудованої системи нечіткого виводу

Після зміни параметрів у полі **Params** для третьої функції приналежності першої змінної на значення [0.385 3.66] одержимо практично точне значення вихідної змінної для значення вхідної змінної 5.29 (рис. 9.1.6).

Більш ефективним способом настроювання, а точніше – модифікації, цієї нечіткої моделі виявляється перший. З цією метою можна збільшити обсяг навчальної вибірки до 12 пар значень, додавши ще три рядки у файл train.txt. Наприклад:

2 1 2.82
3 2 4.12
1 3 2.63

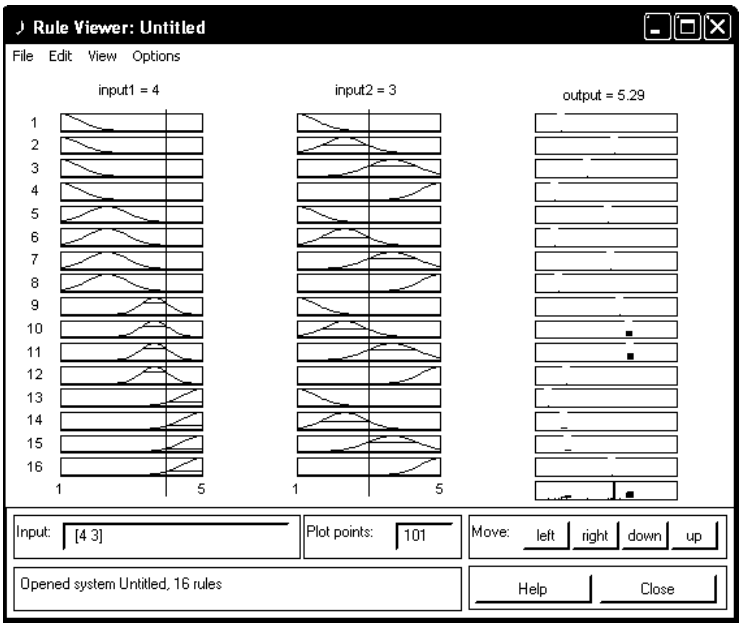

Рис. 9.1.6. Результат ручного настроювання параметрів функції приналежності вхідної змінної для розглянутої системи нечіткого виводу

Після видалення усіх раніше завантажених даних кнопкою **Clear Data** завантажимо новий файл із навчальною вибіркою. При генерації структури нової FIS збільшимо кількість термів і, відповідно, кількість функцій приналежності вхідної змінної до 5, залишивши їх тип без зміни (**gbellmf**). Процес навчання виконаємо аналогічно раніше розглянутому. У результаті буде отримана нова система нечіткого виводу FIS, аналіз якої показує, що, порівняно з першим варіантом нечіткої моделі, вона більш точно описує вихідну математичну функцію. Це може бути основою для генерації нової нечіткої моделі, що пропонується виконати самостійно як вправу.

На закінчення необхідно відзначити, що навіть найпростіші розглянуті приклади відбивають творчий характер процесу побудови й аналізу моделей гібридних мереж. При цьому вибір того чи іншого способу додаткового настроювання нечітких моделей залежить не тільки від специфіки розв’язуваної задачі, але і від обсягу доступної вибірки навчальних і перевірочних даних.

У випадку недостатньої інформації навчальних даних використання гібридних мереж може виявитися взагалі недоцільним, оскільки одержати адекватну нечітку модель, а отже, і точний прогноз значень вихідної змінної не уявляється можливим. Саме з цих причин необхідний попередній аналіз усіх можливостей застосовуваних нечітких моделей для рішення конкретних задач у тій чи іншій проблемній області. Подібний аналіз необхідно виконувати із системної точки зору і з обліком усіх обставин, що складаються на цей момент. Тільки всебічна і повна оцінка проблемної ситуації дозволить розробити адекватну модель розв'язку тієї чи іншої конкретної задачі нечіткого управління чи прийняття рішень.

Завдання для самостійного виконання

Завдання 1. Апроксимація поверхні

За допомогою гібридної мережі провести апроксимацію залежності $y = x^3$, використовуючи дані, які представлені на рис. 9.1.7 та рис. 9.1.8.

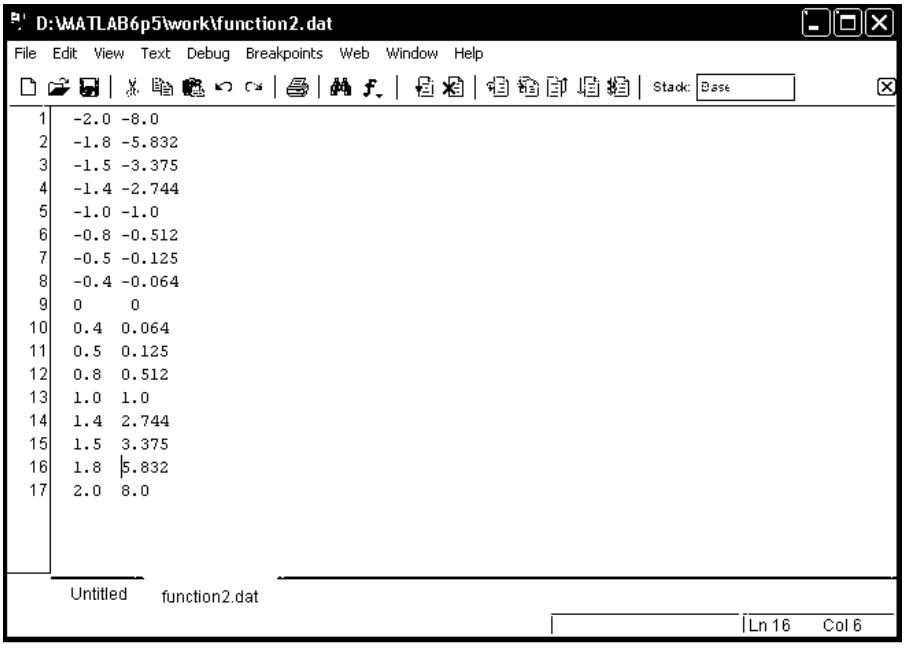

Рис. 9.1.7. Навчальні дані збільшеного об'єму для побудови гібридної мережі ANFIS, що представляє функцію $y = x^3$.

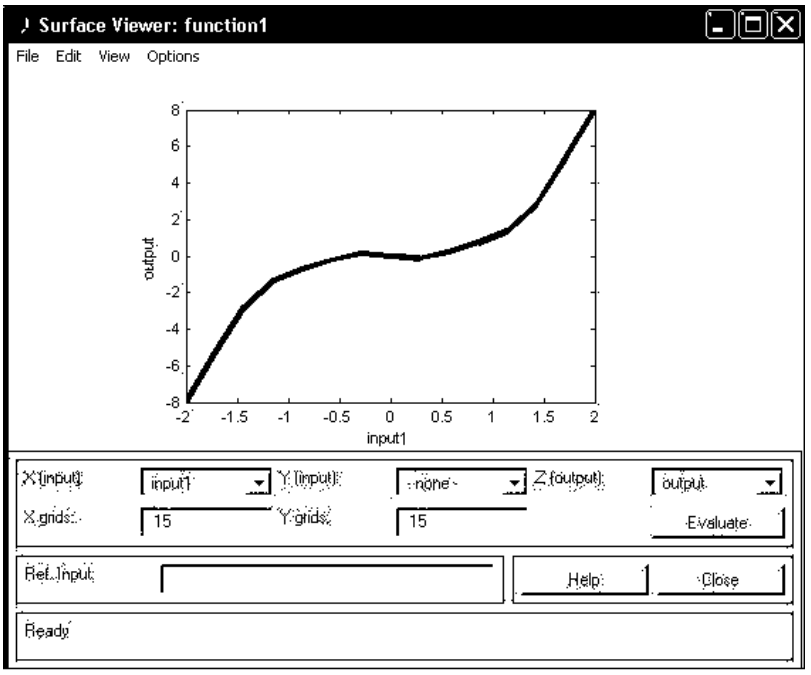

Рис. 9.1.8. Приблизний вигляд поверхні за результатами роботи СНВ, яка апроксимує залежність $y = x^3$

Завдання 2. Аналіз і прогнозування цін на ринку житла

Як приклад побудови і використання адаптивної системи нейронечіткого виводу, розглянемо процес розробки нечіткої моделі гібридної мережі для розв'язування задачі прогнозування ціни на ринку однокімнатних квартир у районі Теремків залежно від кількості метрів та ціни на дану квартиру за попередні три місяці.

Суть цієї задачі полягає в тому, щоб, знаючи динаміку зміни вартості продажу квартир за фіксований інтервал часу, прогнозувати значення їх вартості на визначений момент часу в майбутньому. При цьому характерною рисою динаміки зміни вартості (тренда) є наявність двох основних тенденцій у коливаннях відповідних цін.

З одного боку, спостерігається загальне довгострокове підвищення вартості, пов'язане з інфляцією та іншими факторами. З іншого, – спостерігається короткострокове коливання цін, пов'язане з цілим рядом випадкових факторів, адекватне представлення яких у тій чи іншій формальній моделі навряд чи можливо.

Традиційно для рішення цієї задачі застосовуються різні моделі технічного аналізу, засновані на використанні різних індикаторів. Водночас наявність неявних тенденцій у динаміці зміни вартості житла дозволяє застосувати модель адаптивних нейронечітких мереж.

Як вихідні дані, можна скористатися інформацією про динаміку вартості квартир, які пропонує фірма «Благовест» за 10 останніх тижнів, що доступна в Інтернеті за адресою: **www.blagovest.com**. Цю інформацію для зручності подальшої роботи представимо в табличній формі (табл. 9.1.1).

Таблиця 9.1.1

Динаміка вартості квартир у період з січня по травень N року

Тиждень	**Метраж**	**Вартість**
29.01.N	33	
29.01.N	45	
5.02.N	58	
5.02.N	36	
12.02.N	28	
12.02. N	25	
19.02. N	42	
19.02. N	33	
2.03. N	25	
2.03. N	46	
19.03. N	28	
19.03. N	45	
26.03. N	36	
26.03. N	39	
2.04. N	46	
2.04. N	45	
9.04. N	28	
9.04. N	46	
16.04. N	43	
16.04. N	45	
7.05. N	45	
7.05. N	46	

Припустимо, що нечітка модель гібридної мережі міститиме 3 вхідних змінних. Це будуть значення цін 1 $м^2$ за попередні до поточного 3 тижня. Вихідна змінна – вартість квартири ($1м^2$) на поточний тиждень. Тому у файлі даних слід ввести матрицю цін, яка містить 4 стовпчики. Якщо ми розглядаємо 10 тижнів, тоді кількість рядків буде – 7.

Збережемо навчальну вибірку в зовнішньому файлі під ім'ям prise.txt. Після цього відкриємо редактор ANFIS, у який завантажимо цей файл із навчальними даними.

Перед генерацією структури системи нечіткого виводу типу Сугено після виклику діалогового вікна властивостей задамо для кожної із вхідних змінних по 3 або 4 лінгвістичних терма, а як тип їх функції приналежності виберемо трикутні функції (установлені системою MATLAB за замовчуванням).

Для навчання гібридної мережі скористаємося гібридним методом навчання з рівнем помилки 0, а кількість циклів навчання задамо 20. Після закінчення навчання даної гібридної мережі може бути виконаний аналіз графіка помилки навчання, що показує, що навчання практично закінчилося.

Після навчання гібридної мережі можна візуально оцінити структуру побудованої нечіткої моделі. Очевидно, графічна наочність цієї моделі залишає бажати кращого, оскільки загальна кількість правил у розробленій адаптивній системі нейронечіткого виводу досить багато, що ускладнює їхній візуальний контроль і оцінку.

За допомогою графічних засобів системи MATLAB можна виконати контроль і настроювання параметрів функцій приналежності вхідних змінних і правил нечітких продукцій. Для виконання відповідних операцій можна скористатися редактором функцій приналежності.

Оскільки точність кількісних значень, забезпечувана графічними засобами пакета Fuzzy Logic Toolbox, є недостатньою для вирішення цієї задачі, скористаємося функцією командного рядка evalfis. Як аргументи цієї функції, вкажемо вектор значень вартості

квартири на поточний і 3 попередніх тижні. Повний формат виклику цієї функції буде таким:

out=evalfis ([x1x2 x3], prise),

де out – умовне ім’я вихідної змінної; x1 – значення вартості квартири за попередній тиждень; x2 – значення вартості квартири за два тижні; x3 – значення вартості квартири за три тижні; prise – ім’я структури FIS, попередньо завантаженої в робочу область системи MATLAB.

Після виконання цієї команди за допомогою розробленої нечіткої моделі буде отримане значення вихідної змінної для наступного тижня, буде отримано деяке значення. Порівнюючи отримане значення з відповідним значенням з табл. 2, можна констатувати і порівняти прогноз цін квартир.

У цьому випадку нечіткі моделі адаптивних систем нейронечіткого виводу можуть вважатися новим і конструктивним інструментом технічного аналізу фінансових ринків.

Розглянутий підхід є перспективним напрямом для побудови і використання відповідних нечітких моделей прогнозування цін інших фінансових інструментів, таких як курси валют, акцій компаній, ф’ючерсів і опціонів. Дійсно, загальним для всіх цих інструментів з позицій технічного аналізу є відсутність апріорних припущень про динаміку коливань відповідних курсів цін, що цілком узгоджується з вихідними передумовами побудови нечітких моделей адаптивних систем нейронечіткого виводу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ЗАГАЛЬНА ЛІТЕРАТУРА

1. Матвійчук А. В. Штучний інтелект в економіці : нейронні мережі, нечітка логіка: [монографія] / А. В. Матвійчук. – Київ: КНЕУ, 2011. – 439 с.

2. Гаврилова Т. А. Базы знаний интеллектуальных систем / Т. А. Гаврилова, В. Ф. Хорошевский. – СПб.: Питер, 2001. – 384 с.

3. Леоненков А. В. Нечёткое моделирование в среде MATLAB / А. В. Леоненков. – СПб : БХВ-Петербург, 2003. – 736 с.

4. Штовба С. Д. Введение в теорию нечетких множеств и нечеткую логику [Электронный ресурс] / С. Д. Штовба. – 2002. – Режим доступа: http://www.nsu.ru/matlab/MatLab_RU/fuzzylogic/book1/index.asp.htm

5. Бакан Г. М. Вступ до теорії експертних систем та баз знань / Г. М. Бакан. – К.: ВПЦ «Київський університет», 2005. – 90 с.

6. Дорф Р. Современные системы управления / Р. Дорф, Р. Бишоп; [пер. с англ.]. – М.: Лаборатория Базовых Знаний, 2002 – 832 с.

7. Каллан Р. Основные концепции нейронных сетей / Р. Каллан. – М.: Вильямс, 2001. – 288 с.

8. Ануфриев И. Е. Информатика. Пакет MatLab / И. Е. Ануфриев. – Изд-во СПбГПУ, 2003. – 67 с.

9. Иглин С. П. Математические расчеты на базе MATLAB / С. П. Иглин. – СПб.: БХВ-Петербург, 2005. – 640 с.

ДОДАТКОВА ЛІТЕРАТУРА

Література до теми 1

1. Кудрявцев Д. Многогранность управления знаниями на предприятии [Электронный ресурс] / Д. Кудрявцев. – Режим доступа: // ttp://www.big.spb.ru/publications/bigspb/km/mnogogr_uz_na_predpr.shtml

2. Управление знаниями в корпорациях: учебное пособие / [под ред. Б. З. Мильнера]. – М.: Дело, 2006. – 304 с.

3. Вовк Ю. Я. Підхід до побудови моделі управління знаннями промислового підприємства / Ю. Я. Вовк // Вісник Хмельницького національного університету. – 2009. – № 4. – Т. 2. – С. 217–220.

4. Гаврилова Т. А. Базы знаний интеллектуальных систем / Т. А. Гаврилова, В. Ф. Хорошевский. – СПб: Питер, 2008. – 384 с.

5. Бессмертный И. А. Искусственный интеллект: [учебное пособие] / И. А. Бессмертный. – СПб: СПбГУ ИТМО, 2010. – 132 с.

Література до теми 3

1. Заде Л. Понятие лингвистической переменной и его применение к принятию приближенных решений / Л. Заде. – М.: Мир, 1976. – 165 с.

2. Нечеткая логика: алгебраические основы и приложения: [монография] / С. Л. Блюмин, И. А. Шуйкова, П. В. Сараев, И. В. Черпаков. – Липецк: ЛЭГИ, 2002. – 113 с.

3. Борисов А. Н. Принятие решений на основе нечетких моделей. Примеры моделей / А. Н. Борисов, О. А. Крумберг, И. П. Федоров. – Рига: Зинатне, 1990. – 184 с.

4. Ротштейн А. П. Интеллектуальные технологи идентификации: нечеткая логика, генетические алгоритмы, нейронные сети / А. П. Ротштейн. – Винница: УНІВЕРСУМ – Вінниця, 1999. – 320 с.

5. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилинский, Л. Рутковский. – М.: Горячая линия – Телеком, 2004. – 452 с.

6. Штовба С. Д. Проектирование нечетких систем средствами MATLAB / С. Д. Штовба. – М.: Горячая линия – Телеком, 2007. – 288 с.

Література до теми 9

1. Матвійчук А. В. Штучний інтелект в економіці: нейронні мережі, нечітка логіка: [монографія] / А. В. Матвійчук. – К.: КНЕУ, 2011. – 439 с.

2. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский. – М.: Горячая линия – Телеком, 2004. – 452 с.

3. Штовба С. Д. Проектирование нечетких систем средствами MATLAB / С. Д. Штовба. – М.: Горячая линия – Телеком, 2007. – 288 с.

Навчальне видання

СЕРІЯ «ПОДАТКОВА ТА МИТНА СПРАВА В УКРАЇНІ»

Серію «Митна справа в Україні» засновано в 2001 р. Її продовженням є серія «Податкова та митна справа в Україні», яку започатковано в 2015 р.

Антоненко Володимир Мефодійович,
Мамченко Світлана Дмитрівна,
Рогушина Юлія Віталіївна

СУЧАСНІ ІНФОРМАЦІЙНІ СИСТЕМИ І ТЕХНОЛОГІЇ: УПРАВЛІННЯ ЗНАННЯМИ

Навчальний посібник

Редактор *Н. І. Грицюк, Л. Б. Дьомена*

Форматування та комп’ютерна верстка *О. В. Лисенко*

Здано до друку 25.01.2016. Формат 60×84/16.
Папір офсетний № 1. Гарнітура «Times New Roman».
Друк. арк. 11.1.
Тираж 300 примірників. Замовлення № 541.

Підготовлено до друку Видавничо-поліграфічним центром Національного університету ДПС України
08200, вул. Університетська, 31, м. Ірпінь, Київська область, Україна

Свідоцтво про внесення суб’єкта видавничої справи до державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції
Серія ДК № 4336 від 12.06.2012 р.

У СЕРІЇ «МИТНА СПРАВА В УКРАЇНІ» ВИЙШЛИ ДРУКОМ:

2002

«Основи митної справи»

«Митний контроль та митне оформлення»

2003

«Таможенное оформление морских грузов»

«Основы таможенного дела в Украине»

«Митний контроль на автомобільному транспорті»

2004

«Митний контроль на залізничному транспорті»

«Митне оформлення автотранспортних засобів»

«Основи митної справи в Україні»

«Митний кодекс України та нормативно-правові акти, що регулюють його застосування»

«Митний контроль на повітряному транспорті»

«Митне регулювання зовнішньоекономічної діяльності в Україні»

«Коментар до Митного кодексу України»

2005

«Порядок заповнення вантажної митної декларації»

2006

«Основи мистецтвознавчої експертизи та вартісної оцінки культурних цінностей»

«Історія митної справи в Україні»

«Митне регулювання зовнішньоекономічної діяльності в Україні»

2008

«Основи митної справи в Україні»

2009

«Таможенный контроль: на пути к международным стандартам»

«Таможенный контроль в морских пунктах пропуска»

«Митне оформлення енергоносіїв»

«Митна безпека»

«Міжнародні перевезення товарів автомобільним транспортом: теорія і практика»

2010

«Святий Матфей. Митар. Апостол. Євангеліст»

«Історія митної діяльності: Україна в європейському контексті»

2011

«Митні інформаційні технології»

«Особливості митної класифікації товарів»

«Інфраструктура відкритих ключів: технології, архітектура, побудова та впровадження»

«Розподілені бази даних»

2012

«Інформаційно-аналітичне забезпечення діяльності митних органів»

«Митний кодекс України» / Коментар до ст. 6»

«Митна політика та митна безпека: концептуальне визначення та шляхи забезпечення»

«Архітектура комп'ютера»

«Комп'ютерна схемотехніка»

2013

«Актуальні питання теорії та практики митної справи»

«Історія митної діяльності»

«Механізми державного управління митною справою»

«Формування системи митного аудиту»

2014

«Митна енциклопедія, т. 1»

«Митна енциклопедія, т. 2»

«Митна енциклопедія, т. 3»

СЕРІЮ «МИТНА СПРАВА В УКРАЇНІ» ЗАСНОВАНО В 2001 Р. ЇЇ ПРОДОВЖЕННЯМ Є СЕРІЯ «ПОДАТКОВА ТА МИТНА СПРАВА В УКРАЇНІ», ЯКУ ЗАПОЧАТКОВАНО В 2015 Р.

У СЕРІЇ «ПОДАТКОВА ТА МИТНА СПРАВА В УКРАЇНІ» ВИЙШЛИ ДРУКОМ:

2015

«Реформування податкової системи України: сучасні виклики та орієнтири»
«Міжнародні економічні відносини: у 2-х частинах»
«Адміністративно-правове сприяння соціально-економічному розвитку в Україні: теоретико-методологічні засади»
«Бухгалтерський облік в умовах застосування інформаційних технологій: історія та практика»
«Розслідування злочинів, пов'язаних із неналежним виконанням професійних обов'язків медичними працівниками: сучасний стан, проблеми та напрями удосконалення»
«Формування управлінської компетентності майбутніх економістів у процесі професійної підготовки: теорія і практика»
«Let Develor English skills»
«Кримінальний процес України»
«Європейська інтеграція»
«Проблеми правового закріплення принципів фінансового контролю»
«Протидія розслідуванню кримінальних правопорушень у сфері оподаткування: теорія та практика»
«Інформатика та обчислювальна техніка: практикум»

2016

«Регулятивний потенціал фінансового ринку в умовах глобальних викликів»
«Правовий статус фіскальних органів України та держав-членів Європейського Союзу»
«Психологічні основи у професійній підготовці фахівців ПМ»
«Сучасні інформаційні системи і технології: управління знаннями»
«Економічна кібернетика: введення в спеціальність»
«Інформаційні системи та технології у фінансових установах»
«Фінансове забезпечення інноваційного розвитку»
«Практикум з інформаційних систем в економіці»
«Практикум з WEB-програмування»
«Малий та середній бізнес у взаємовідносинах із податковими органами: соціологічна інтерпретація»
«Моделювання систем захисту інформації»
«Формування бюджетної стратегії України: теорія, методологія, практика»
«Актуальні питання розвитку банківської системи України»
«Система державного фінансового контролю: формування та розвиток в Україні»
«Соціалізація студентської молоді у процесі професійної підготовки»

2016
«Правове забезпечення фінансової безпеки України»
«Розвиток бухгалтерського обліку в умовах інтеграційних процесів»
«Алгоритмізація та програмування»
«Основи психології та конфліктології для фахової підготовки економістів»
«Інформаційні системи і технології»
«Інформаційні системи і технології на підприємстві»
«Проблеми соціологічного, психологічного та педагогічного забезпечення податкових відносин в Україні»
«Теорія ймовірностей та математична статистика: практикум»
«Математика для економістів. Функція багатьох змінних. Ряди»
«Математика для економістів. Вища математика. Частина І. Лінійна алгебра. Аналітична геометрія»
«English grammar for low students»
«Grammar for students of economics»

В УНІВЕРСИТЕТІ ДЕРЖАВНОЇ ФІСКАЛЬНОЇ СЛУЖБИ УКРАЇНИ ВИХОДИТЬ СЕРІЯ «ПОРТРЕТИ ВЧЕНИХ УНІВЕРСИТЕТУ ДФС УКРАЇНИ», ЗАСНОВАНА 2012 РОКУ, А ТАКОЖ СЕРІЯ «НА ДОПОМОГУ СТУДЕНТУ УДФСУ», ЗАПОЧАТКОВАНА 2016 РОКУ.