Міністерство освіти і науки України

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Л. М. Добровська

НЕЧІТКІ МОДЕЛІ В МЕДИЦИНІ

КОМП’ЮТЕРНИЙ ПРАКТИКУМ

Рекомендовано Методичною Радою КПІ ім.Ігоря Сікорського як навчальний посібник для студентів, які навчаються за спеціальністю 122 “Комп’ютерні науки”

Київ
КПІ ім.Ігоря Сікорського
2019

Рецензенти:
Жданова О. Г., к.т.н., доц. каф. АСОІУ ФІОТ НТУУ «КПІ ім.Ігоря Сікорського»
Антонова-Рафі Ю. В. , к.т.н., доц. кафедри БЗЛ ФБМІ НТУУ «КПІ ім. Ігоря Сікорського»

Відповідальний редактор Носовець О. К., к.т.н.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 8 від 25.04.2019 р.)
за поданням Вченої ради Факультету біомедичної інженерії
(протокол № 9 від 25.03.2019 р.)

Електронне мережеве навчальне видання

Добровська Людмила Миколаївна, к. пед. н.

НЕЧІТКІ МОДЕЛІ В МЕДИЦИНІ

КОМП'ЮТЕРНИЙ ПРАКТИКУМ

Нечіткі моделі в медицині. Комп'ютерний практикум
[Електронний ресурс] : навчальний посібник для студентів спеціальності 122 "Комп'ютерні науки" для всіх спеціалізацій / Л. М. Добровська; КПІ ім. Ігоря Сікорського. – Електронні текстові данні (1 файл: 23,1 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2019 – 315 с.

Посібник призначений для викладачів, які проводять комп'ютерний практикум з дисципліни «Нечіткі моделі в медицині» (або «Методи та технології обчислювального інтелекту»), та студентів спеціальності 122 "Комп'ютерні науки". У посібнику розкриваються методичні та технологічні засади створення нечітких моделей, гібридних моделей з елементами нечіткості з прикладами розв'язання типових завдань, що при успішному виконанні і засвоєнні матеріалу сприятиме формуванню вмінь проектувати, розробляти та використовувати ці сучасні інформаційні технології при проектуванні компонентів відповідного програмного забезпечення, набуттю здатностей до застосування теоретичних знань в практичній діяльності.

За редакцією укладача

З М І С Т

Умовні скорочення

ГА – генетичний алгоритм,

НМ – нейронна мережа,

СНВ – система нечіткого виведення,

ФН – функція належності,

ФП – функція пристосованості,

ЦФ – цільова функція.

ВСТУП

У наш час прискореними темпами розвиваються IT-технології, що об'єднуються в англомовній літературі під назвою *Computational Intelligence*. Вони дозволяють отримувати неперервні або дискретні рішення в результаті навчання за доступними наявними даними. В цьому плані це видання дуже актуально, оскільки сприяє ефективному оволодінню новими технологіями у вигляді *інтелектуальних обчислювальних систем*. Останні є об'єднання нейронних мереж, генетичних алгоритмів і нечітких систем, взаємодія яких дозволяє вирішувати різні завдання, але найважливіше – вони стають універсальним інструментом обробки інформації.

Дисципліна «Нечіткі моделі в медицині» досліджує інтелектуальні методи вирішення різних завдань на основі нечітких моделей та відповідних гібридних моделей, які знаходить застосування в медицині. Книга містить базові елементи та приклади реалізацій конкретних технічних рішень, отриманих на основі нечітких систем, деяких нейронних мереж і генетичних алгоритмів (в тому числі різних систем, які навчаються та засновані на нечіткій логіці).

Проблематика нейронних мереж, генетичних алгоритмів і нечітких систем, особливо комбінація цих методів – це одна з областей досліджень, яка найбільш інтенсивно розвивається в наш час та отримала назву «Обчислювальні технології». Її можна вважати сучасним відгалуженням інформатики (Computer Science), пов'язаним із методами штучного інтелекту (Artificial Intelligence).

Самостійна робота студента є основною складовою навчальних занять з курсу «Нечіткі моделі в медицині». Однак організація самостійної роботи вимагає певної методичної підготовки, що передує кожному заняттю. Завдання методичних вказівок – акцентувати увагу студентів на основних проблемах предмета й тим самим сприяти більш глибокому засвоєнню вивченого матеріалу.

Заняття з курсу «Нечіткі моделі в медицині» повинні супроводжуватися апробацією вивчаємого матеріалу шляхом використання ЕОМ. У зв'язку з цим основним темам курсу відповідають індивідуальні завдання, окремі з яких, на розсуд викладача, можуть бути виконані групою студентів.

Інструментальними засобами виконання завдань є програмні середовища MatLab (разом із пакетом Fuzzy Logic Toolbox) та Python. Тому навчальний курс передбачає виконання індивідуальних завдань, пов'язаних із програмуванням в названих середовищах, проектуванням та дослідженням систем нечіткого логічного виведення та гібридних систем.

Виконання практичної роботи передбачає виконання таких кроків:

1. Вивчити теоретичні відомості.
2. Послідовно виконати всі завдання до практичної роботи.
3. Перевірити правильність виконання завдання.
4. Оформити звіт.
5. Відповісти на контрольні запитання.

Звіт з виконаної практичної роботи має складатися з таких структурних елементів: 1) титульний лист; 2) завдання (постановка задачі); 3) алгоритм виконання роботи; 4) одержані результати; 5) висновок; 6) додаток «Лістинг програми».

Розділ 1. Основні поняття теорії нечітких множин та нечіткої логіки

Тема 1. Організація обчислень і програмування в системі MatLab

1. Загальна характеристика системи MatLab. Організація обчислень. Основи програмування в системі MatLab

Мета роботи: ознайомитися із: 1) загальними поняттями системи MatLab, 2) особливостями програмування в середовищі MatLab.

Об'єкт дослідження – система MatLab.

Питання для опрацювання

1. Основні прийоми роботи у вікні команд [3, с. 6–12].
2. Графічні можливості системи MatLab [3, с. 13–15].
3. Основи програмування в системі MatLab: m-файли в системі MatLab, виконання m-функцій, синтаксис основних операторів MatLab [3б с. 16–26].

1. Контрольні запитання та завдання

1. Назвіть основні прийоми роботи у вікні команд системи.
2. Охарактеризуйте графічні можливості системи MatLab.
3. Що ви знаєте про організацію обчислень в системі.
4. Наведіть приклади допустимих операцій над матрицями даних.
5. Назвіть основні ознаки, які відрізняють m-сценарій від m-функції. Наведіть приклади оформлення заголовків *m*-функцій.

Аудиторна робота

1. Сформувати матрицю – тригонометричну таблицю, першим стовпчиком якої є значення аргумента (кута, який змінюється в інтервалі від 0^0 до 360^0 з кроком 30^0), а другим і третім – значення функцій *sin, cos* від даного аргумента.

2. Побудувати графіки функцій: а) $y=2^x$; $x\in[0, 30]$; б) $z=cos(x, y)$, $x\in[0^0, 360^0]$.

Теоретичні відомості

1. Основні прийоми роботи у вікні команд [3]

Система MatLab є інтегрованим програмним середовищем для виконання чисельних розрахунків, комп'ютерного моделювання та обчислювальних експериментів, які охоплюють різні області математики. Система MatLab складається з базової програми та декількох десятків пакетів (Toolbox), які забезпечують розв'язання системою завдань різної складності. В результаті запуску системи MatLab на екрані монітору з'явиться робочий інтерфейс

програми MatLab (рис. 1.1). При цьому кожний запуск системи називається *сеансом роботи* з системою.

Панелью доступу до компонентів системи є меню Start (рис. 1.2), організоване у вигляді меню швидкого запуску «Пуск». Ця панель призначена для виклику потрібної довідки та спеціальних графічних засобів, перегляду демонстраційних прикладів тощо. Основну частину вікна MatLab займає вікно команд (рис. 1.3), призначене для взаємодії користувача з системою в *режимі командного рядка*. Вікно команд використовують для введення команд і функцій з необхідними аргументами, для присвоєння змінним деяких значень і відображення результатів виконаних обчислень.

Застосування спеціальних графічних інтерфейсів користувача GUI (Graphic User Interface) для розв'язання окремих класів задач називають *графічним* або *інтерактивним режимом* роботи. Розглянемо вкладки, розташовані в лівій верхній частині головного вікна системи MatLab: вікно поточного каталогу або папки системи (Current Directory) та вікно перегляду робочої області системи (Workspace).

Вікно Workspace (рис. 1.4) надає можливість переглянути вміст робочої області та редагувати значення окремих змінних. При цьому варто пам'ятати, що всі змінні система розглядає як матриці або масиви. Тому у вікні перегляду робочої області вказується не тільки ім'я змінної, але й її розмір, об'єм пам'яті, який вона займає (в байтах), і тип цієї змінної.

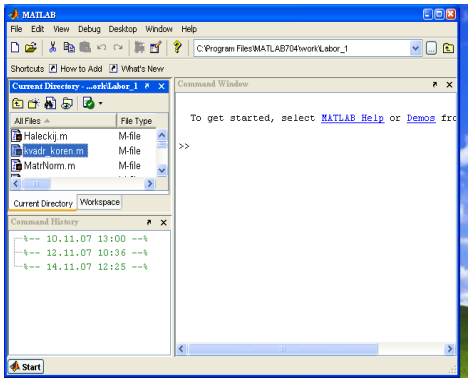


Рис. 1.1. Загальний вигляд графічного інтерфейсу системи

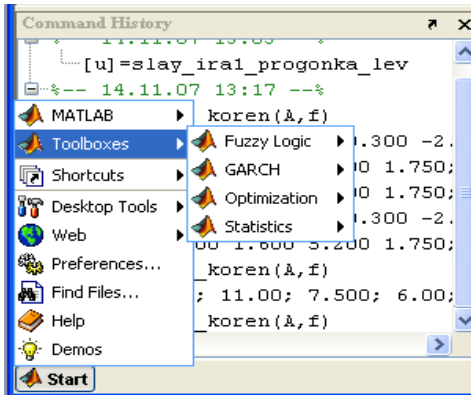


Рис. 1.2. Вікно доступу до компонентів системи (меню Start)

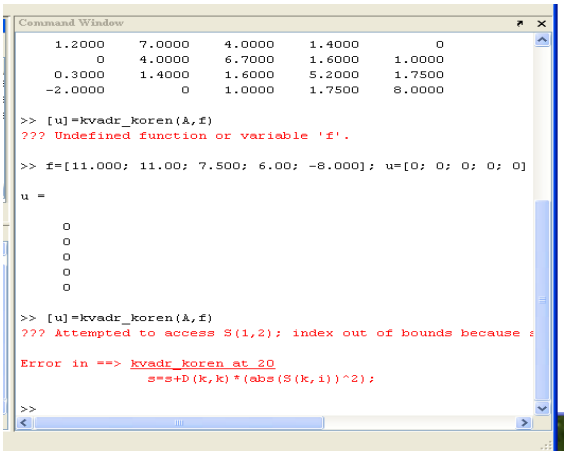


Рис. 1.3. Вікно команд системи

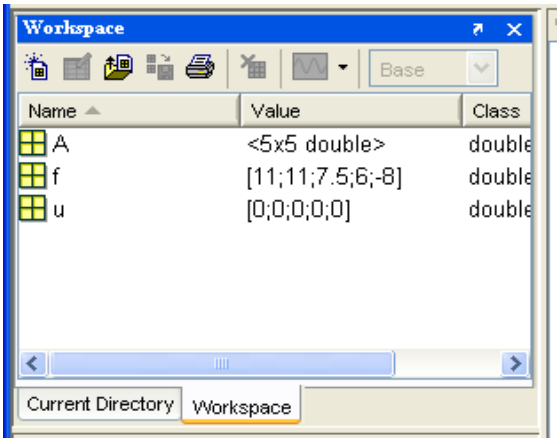


Рис. 1.4. Вікно перегляду робочої області системи

Вікно Workspace дозволяє видалити обрану змінну, очистити всю робочу область, завантажити змінні в робочу область із зовнішнього файлу та зберегти значення змінних в зовнішньому файлі.

Вікно поточного каталогу (рис. 1.5) відображає файли обраного каталогу системи MatLab. Файли з розширеннням *m*, *dat*, *fis* є звичайними текстовими файлами, які можна переглядати й редагувати у будь-якому ASCII-редакторі. Однак, більш зручним для цієї мети є вбудований редактор системи MatLab – редактор налагодження *m*–файлів Editor.

У лівій нижній частині головного вікна системи MatLab розташована вкладка, яка містить вікно історії команд Command History (рис. 1.6). Це вікно відображає всі команди, введені користувачем як під час поточного сеансу роботи із системою, так і під час попередніх сеансів.

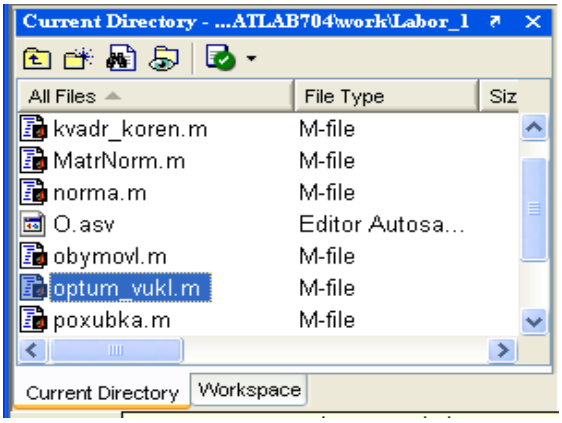


Рис. 1.5. Вікно поточного каталога системи MatLab

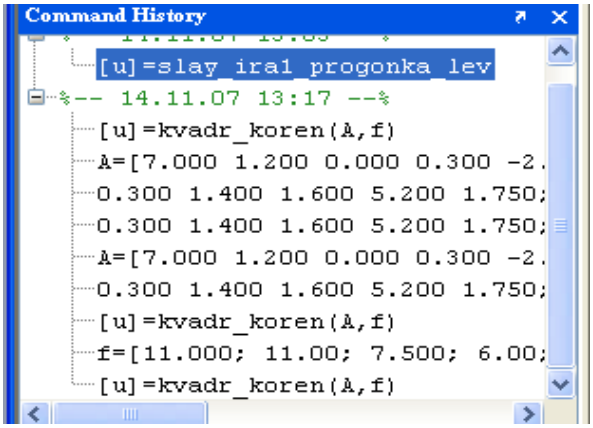


Рис. 1.6. Вікно історії команд

Для виконання однієї з цих команд достатньо двічі натиснути на її ім’я. Для редагування обраної команди її потрібно перемістити у вікно команд. Переглянути раніше виконані команди можна також скориставшись клавішами «↑» і «↓» у вікні команд.

Основні прийоми роботи у вікні команд. Введення команди здійснюється набором відповідних символів у окремому рядку активного вікна команд після символів запрошення «>>». Аналогічно вводяться й окремі значення змінних, при цьому після імені змінної необхідно поставити символ «=», який в системі відіграє роль знака присвоювання. Символи верхнього і нижнього регістрів сприймаються системою як різні. Якщо ім’я команди або значення змінної занадто довге, то варто використати символ продовження введення «...». При цьому всі набрані в новому рядку символи вважаються продовженням символів попереднього рядка.

Якщо після введення імені команди (або значення змінної) поставлено символ «;», то результат її виконання у вікні команд не відображається.

Якщо результат виконання окремої команди набуває деякого значення, яке не присвоєно жодній змінній, то це значення присвоюється, за замовчуванням, змінній з ім’ям *ans*. Тип кожної змінної визначається її значенням.

Масиви і матриці є базовими об’єктами мови MatLab. З часом ця мова поповнювалася новими об’єктами (наприклад, об’єктами *net* для опису нейронних мереж, *fis* для опису систем нечіткого логічного виведення).

Для визначення значень елементів матриці використовуються символи: «[» – для початку і «]» – для закінчення окремої матриці. При цьому матриці записуються по рядках: елементи матриці відокремлюють один від одного прогалиною, а рядки – символом «;». При введенні матриць розміром 1×1 знаки початку та закінчення матриці можна опустити. Наприклад, якщо необхідно присвоїти змінній з ім’ям X значення 10, то у вікні команд після символів «>>»

треба ввести такий рядок: X=10. Аналогічно можна присвоїти значення рядка деякій змінній з ім’ям b:

```
>> b='String'
b = String
```

Щоб задати матрицю **M** вигдяду: $\mathbf{M} = \begin{bmatrix} 1 & 0.1 & 0.2 \\ 0.8 & 0.9 & 1 \\ 0.7 & 0.8 & 0.5 \\ 1 & 0.5 & 0.2 \end{bmatrix}$ необхідно ввести її значення таким чином: M=[1 0.1 0.2; 0.8 0.9 1; 0.7 0.8 0.5; 1 0.5 0.2]

До цієї матриці можуть бути застосовані різні унарні функції (транспонування, знаходження оберненої матриці, мінімальних і максимальних елементів у кожному стовпчику матриці тощо). Наприклад, для знаходження мінімальних елементів у кожному стовпчику матриці **M** необхідно скористатися функцією *min*:

```
>> min (M)
ans = 0.7000   0.1000  0.2000
```

Для звернення до окремих елементів матриці треба після її імені вказати в круглих дужках індекси відповідного елемента: номер рядка та номер стовпчика, які розділені комами (наприклад, M(1, 2)).

У деяких випадках для присвоєння значень елементам масивів і матриць зручним є використання оператора *двокрапка* («:»), дія якого аналогічна присвоєнню значень змінній в циклі типу *for*. При цьому форма запису має вигляд: «*a*:*b*:*c*», де *a* – значення першого елемента масиву, *b* – крок зміни елементів, *c* – значення останнього елемента масиву. Наприклад, при побудові графіків функцій необхідно визначити діапазони значень відповідних змінних. Це зручно зробити таким чином: X=0:0.1:10. В результаті введення цього рядка буде визначено масив *X*, який складається зі 101 елемента, першим з яких є число 0, а останнім – число «10».

Для введення матриць в системі передбачені окремі функції, деякі з яких наведено в табл. 1.1. Наприклад, щоб визначити матрицю *Z* розміром 3×4, яка

складається з нулів, у вікні команд потрібно ввести ім'я відповідної функції *zeros* та її параметри: Z=zeros(3, 4). При написанні алгебраїчних виразів MatLab дозволяє використовувати традиційні знаки арифметичних операцій. Перелік основних допустимих операцій над масивами і матрицями наведено в табл. 1.2.

Таблиця 1.1

Найпростіші функції для введення матриць

Ім'я функції	Призначення функції
ones	введення матриці, яка складається з одиниць
zeros	введення матриці, яка складається з нулів
rand	введення матриці, елементи якої розподілені за нормальним законом на проміжку [0, 1]
randn	введення матриці, елементи якої розподілені за нормальним законом
eye	введення матриці, головна діагональ якої складається з одиниць
magic	введення "магічної" матриці, сума елементів якої по рядкам, по стовпчикам і по діагоналі є однаковою.

Таблиця 1.2

Перелік основних операцій над масивами і матрицями

Операції	Масив	Матриця
Вихідний масив або матриця	$A=[a_{ij}]$	$A=[a_{ij}]$
Додавання/віднімання	$A \pm B=[a_{ij} \pm b_{ij}]$	$A \pm B=[a_{ij} \pm b_{ij}]$
Добуток	$A.* B=[a_{ij} * b_{ij}]$	$A* B=[A(i,:)*B(:,j)]=\left[\sum_{k=1}^{p} a_{ik} b_{kj}\right]$
Ліве ділення	$A./ B=[a_{ij} / b_{ij}]$	$B/A=BA^{-1}$
Праве ділення	$A.\backslash B=[b_{ij} / a_{ij}]$	$A\backslash B=A^{-1} B$
Транспонування	$A.'=[a_{ji}]$	$A'=[a^{*}_{ji}]$, де "*" – операція комплексного спряження
Піднесення до степеня	$A.\hat{} B=[a_{ij} \hat{} b_{ij}]$	A^k=A*…*A

Тут символ «.» вказує на те, що виконується поелементна операція.

Такі традиційні операції, як додавання, у виразах MatLab виконуються особливим способом. Найбільш звичним є додавання скалярних величин (тобто масивів розміру 1×1), яке відповідає аналогічній дії в більшості алгоритмічних мов.

Якщо обома операндами є масиви однакового розміру, то здійснюється покомпонентне додавання елементів з однаковими індексами, наприклад,

A=B+C. Однак, якщо в MatLab до масиву будь-якого розміру додається скалярна величина, то вона додається до кожного елемента масиву. При спробі скласти масиви різного розміру (за винятком випадку, коли один із операндів є масивом 1×1) система видає відповідне повідомлення про помилку.

При виконанні логічних операцій над числами діє наступна умова: ненульові значення розглядаються як істина, а нульові – як хибність.

Інформація про всі функції системи MatLab міститься в довіднику системи.

2. Графічні можливості системи MatLab [3]

Система має зручні та потужні засоби візуалізації та графічного зображення різноманітних математичних об'єктів типу кривих, поверхонь і діаграм на площині та у 3-вимірному просторі. При цьому використовуються різні системи координат, стилі та способи виділення кольором зображень.

Розглянемо основні функції системи, які можуть бути використані для візуалізації окремих властивостей нечітких моделей.

Для побудови графіка функції однієї змінної насамперед необхідно визначити множину (масив) значень незалежної змінної та відповідну множину значень залежної (функціональної) змінної. Нижче наведено послідовність команд для побудови графіка функції $y=sin(x)$ за допомогою функції *plot:* X=-10:0.1:10; Y=sin(X); plot(X,Y).

Після виконання даної послідовності команд виникне нове вікно з графіком цієї функції (рис. 1.7), яке має власне головне меню та панель команд. Відповідні команди дозволяють виконати редагування властивостей отриманого графіка, наприклад, внести додатковий текст (легенду), змінити колір, тип ліній і масштаб зображення, виконати поворот зображення.

Для побудови графіків 3-вимірних поверхонь і кривих, які є зображеннями функцій двох змінних, можна скористатися такими функціями системи MatLab: 1) *plot3* – функція побудови зображень 3-вимірних поверхонь лініями; 2) *mesh* – функція побудови зображень 3-вимірних поверхонь із

функціональним розфарбовуванням ліній контурної сітки; 3) *surf* – функція побудови зображень 3-вимірних поверхонь з функціональним розфарбовуванням комірок контурної сітки.

При цьому для присвоєння значень незалежним змінним зручно скористатися функцією meshgrid: [*X*,*Y*]=*meshgrid*(*x*,*y*), яка перетворює область, визначену векторами *x* і *y*, в масиви X і Y. Нижче наведено послідовність команд, необхідних для побудови графіка поверхні *z*=*min*(*x*,*y*) за допомогою функції *surf*() (рис. 1.8):

```
>> [x,y]=meshgrid([-1:0.1:1]); z=min(x,y); surf(x,y,z)
```

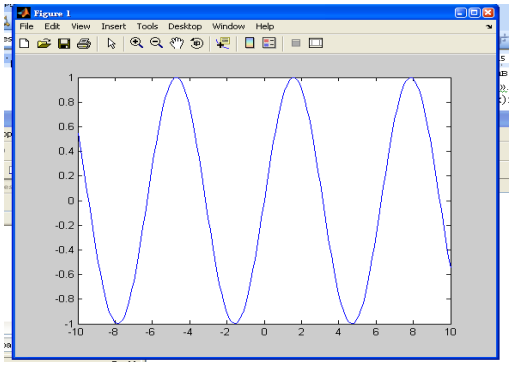


Рис. 1.7. Вікно із побудованим графіком функції *y = sin(x)*

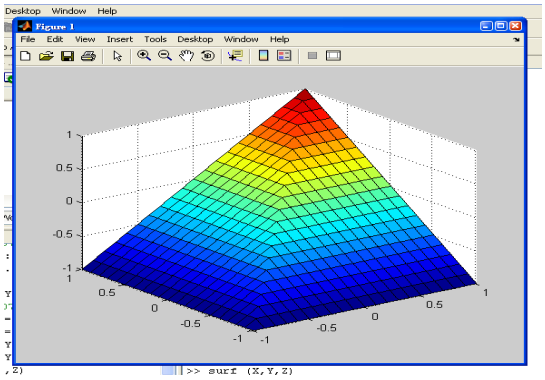


Рис. 1.8. Поверхня z=min(x, y), побудована за допомогою функції *surf*

Наведемо перелік деяких функцій, які можна використати для побудови зображень: 1) графіків кривих і поверхонь – функції *meshc, meshz, surfc* тощо; 2) діаграм у вигляді стовпчиків – функції *bar, barh, hist*; 3) кругових діаграм – функції *pie* (плоска) та *pie3* (3-вимірна).

3. Основи програмування в системі MatLab [3]

M-файли в системі MatLab (сценарії та функції). Файли, які містять команди та оператори системи MatLab, називаються *m-файлами.* Розрізняють два типи m-файлів: *сценарії* (m-сценарії) та *функції* (m-функції). Вони можуть бути створені в будь-якому текстовому редакторі та збережені у файлі з розширенням *m*. Однак система MatLab має особистий редактор m-файлів – *Editor*, який можна викликати:

1) за допомогою опції New→M-File з меню File;

2) з командного рядка системи MatLab, ввівши команду *edit*;

3) за допомогою меню швидкого запуску «Пуск».

Зазвичай, при написанні m-функцій ім'я m-файлу, в якому запам'ятовується програма, співпадає з им'ям функції.

М-сценарії дозволяють автоматизувати виконання послідовності операторів. В них можна звертатися до будь-яких команд та функцій системи MatLab, при цьому не потрібно оголошувати імена та типи змінних. Сценарії можуть виконуватися як безпосередньо з файлу, так і шляхом копіювання всього сценарію або його фрагментів у командне вікно. Вони оперують даними з робочої області Workspace й можуть створювати нові дані з метою їхньої подальшої обробки. Декілька сценаріїв можуть виконуватися послідовно один за одним, передаючи дані наступному сценарію через робочий простір або запам'ятовуючи проміжні результати в файлах. Один m-файл може включати в себе декілька m-сценаріїв.

Сценарій, зазвичай, розпочинається із заголовку *script*, за яким розміщено рядок коментарю, який пояснює призначення сценарію. Коментар може приймати вигляд окремого рядка або розміщуватися після будь-якого оператора, за умови, що йому передує символ «%». Наведемо приклад сценарію (файл *exemple.m*), який обчислює радіус-вектор *rho* для тригонометричної функції *sin* в залежності від кута *theta* та будує відповідний графік в полярних координатах:

```
Script
%M-file – сценарій побудови графіків в полярних координатах
theta=-pi:0.01:pi;
rho(1,:)=2*sin(5*theta).^2;
polar(theta,rho(1,:)) % побудова графіка
```

Введення команди *exemple* у вікні команд призводить до виконання цього сценарію. В результаті буде побудовано графік, зображений на рис. 2.1. В

сценарії відсутні вхідні та вихідні аргументи: програма сама створює змінні, які зберігаються в робочій області системи. Коли виконання сценарію завершено, змінні *theta* та *rho* залишаються в робочій області. Для того, щоб переглянути їх, треба відкрити вікно робочої області Workspace або скористатися командою *whos*. Запуск сценаріїв на виконання може також здійснюватися і в середовищі текстового редактора Editor: 1) за допомогою опції Run меню Debug (або клавіші F5), 2) за допомогою опції Evaluate Selection меню Text.

Команда Evaluate Selection забезпечує обчислення виділеного фрагменту m-сценарію, якщо в поточний момент відомі значення всіх змінних, які входять до нього.

Результат даних обчислень відображається у вікні Command Window.

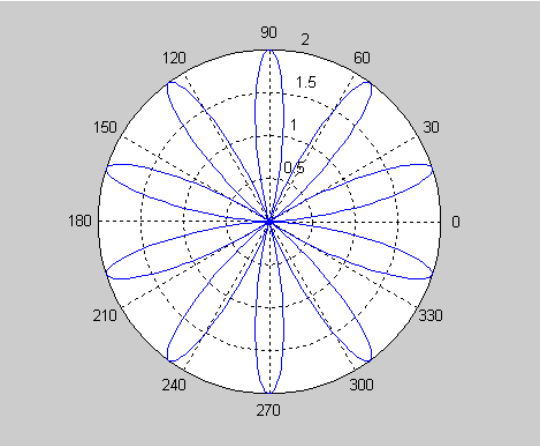


Рис.1.9. Результат виконання команди *exemple*

М-функції дозволяють створювати нові бібліотеки m-функцій і пакети прикладних програм. Вони, на відміну від сценаріїв, мають вхідні та вихідні аргументи та створюють локальні змінні в робочій області m-функції, яка відрізняється від робочої області системи MatLab.

Структура m-функції характеризується наступними компонентами:

1. *Рядок визначення функції* (*заголовок функції*) задає її ім’я та послідовність вхідних і вихідних аргументів (змінних).

Можливі такі варіанти оформлення заголовків функцій:

function *f*1 function *y*=*f*3(*x*l, *x*2,...)

function *f*2(*x*l, *x*2,...) function [*y*l, *y*2,...]=*f*4(*x*l, *x*2,…),

де *f*1, *f*2, *f*3, *f*4 – заголовки функцій, (*x*l, *x*2, ...) – вхідні аргументи; (*y*l, *y*2, ...) – вихідні аргументи. Якщо m-функція має більше одного вихідного аргумента, то вони розміщуються в квадратних дужках і відокремлюються один від одного

комами. Вхідні аргументи m-функції, якщо їх більше одного, розміщуються в круглих дужках.

2. *Перший рядок коментарю,* зазвичай, визначає призначення функції. Він використовується командою *he1p <ім'я каталогу>* для виведення на екран загальної інформації про всі m-функції заданого каталогу.

3. *Тіло функції* – це програмний код на мові MatLab, який реалізує обчислення та присвоює значення вихідним змінним, наприклад:

```
Заголовок функції:        function f=factorial (n)
Перший рядок коментарю:   % FACTORIAL – обчислення факторіалу
Коментар:                 % factorial(n) визначає значення n!
Тіло функції:             f=prod(1:n);
```

М-файли можуть містити коди декількох функцій. Перша функція в файлі – *основна функція*, яка викликається за ім'ям файлу. Інші функції являють собою *підфункції*, які доступні лише основній функції та іншим підфункціям цього файлу.

Кожна підфункція має свій власний заголовок. Підфункції розташовуються одна за одною й можуть викликатися в довільному порядку в тілі основної функції, наприклад:

```
Основна      function [avg,med]= newstats(u)
функція      % NEWSTATS знаходить середнє значення та медіану
             % для елементів вектора u з використанням підфункцій
             n=length(u);
             avg=mean_a(u,n);
             med=median_a(u,n);
Підфункції   function m=mean_a(v,n)
             % ОБЧИСЛИТИ середнє
             m=sum(v)/n;
             function m=median_a(v,n) % ОБЧИСЛИТИ медіану
              w=sort(v);
              if rem(n,2)==1
                  m=w((n+1)/2);
              else m=(w(n/2)+w(n/2+1))/2;
             end
```

MatLab дозволяє використовувати «анонімні» функції, які не мають особистого імені, а замість нього використовують вказівку на функцію. Формат оголошення таких функцій приймає наступний вигляд:

h_F=@ (список_параметрів) формула

Функції такого типу корисно застосовувати при роботі з формулами. Наприклад, “анонімна” функція піднесення до квадрату з вказівкою *sqr* може бути описана та використана наступним чином:

```
>>sqr=@(x)x.^2; sqr([1 2 3])
ans = 1 4 9
```

Вказівка на анонімну функцію може бути передана іншим функціям як параметр. Наприклад, ми можемо звернутися до функції обчислення визначеного інтеграла від x^2 на інтервалі від 0 до 1 наступним чином:

```
>> quad (sqr, 0, 1)
ans = 0.3333
```

М-функції викликаються аналогічно вбудованим функціям системи MatLab. Їх також можна, на відміну від сценаріїв, застосовувати в арифметичних виразах.

Розглянемо m-файл, який обчислює середнє значення елементів вектора.

```
function y=average(x)
% AVERAGE(x) обчислює середнє значення елементів вектора
% Якщо вхідний аргумент не є вектором, генерується помилка
[m,n]=size(x);
if (~((m==1)|(n==1))|(m==1&n==1))
 error('Вхідний масив повинен бути вектором')
end
y=sum(x)/length(x);
```

Оператори в тілі функції можуть складатися з виклику функцій, програмних конструкцій для управління потоком команд, інтерактивного вводу-виводу, обчислень, операторів присвоєння, коментарів і порожніх рядків. Звернутися до функції *average* можна таким чином:

```
>>y=average(1:99)
y = 50
```

Серед особливостей написання програм мовою MatLab можна виділити такі:

1. Кожний оператор програми рекомендовано записувати в окремому рядку.

2. При необхідності можна використовувати оператори, розташовані на декількох рядках, при цьому рядки завершуються символом продовження “...”.

3. При створенні m-функції дозволяється розміщувати в одному рядку декілька операторів, за умови, що вони відокремлені один від одного символом «,» або «;».

4. При написанні m-функцій, з метою запобігти виведенню на екран проміжних результатів, всі оператори мають завершуватися символом “;”.

5. Послідовність символів, яка розпочинається з символу *“%”,* не виконується та сприймається системою як коментар.

6. Рядки, які передують першому оператору програми, сприймаються системою як коментар.

7. Змінні, як правило, не описуються і не оголошуються. Будь-яка нова змінна сприймається системою як ім’я масиву визначеного класу.

8. Імена змінних можуть включати лише букви латинського алфавіту, цифри й знак підкреслювання та завжди розпочинаються з букви. Довжина імені не обмежена, але система MatLab розрізняє тільки перші 63 символи.

Виконання М-функцій. Система MatLab виконує m–файли в режимі інтерпретації. При цьому вона, зазвичай, використовує проміжний *псевдокод* (p-code), який дозволяє звести до мінімуму роботу щодо синтаксичного аналізу операторів програми. Під час виклику m-функції система перетворює функцію у псевдокод і завантажує в оперативну пам’ять. Псевдокод залишається в пам’яті, доки не буде завершено сеанс роботи із системою або застосована одна із команд *clear:* 1) clear <ім’я_функції> – видалення конкретного псевдокоду; 2) clear functions – видалення псевдокодів всіх функцій; 3) clear all – видалення всіх функцій і змінних. Можна також зберігати псевдокод m–функції з метою

використання його в наступних сеансах роботи. Для цього застосовують команду: pcode <ім'я файлу>, яка виконує синтаксичний аналіз m-файлу, створює псевдокод і зберігає його у файлі з тим же ім'ям, але з розширенням *p*.

Розглянемо більш детально процес виконання М-функцій.

Виклик m-функції. М-функцію можна викликати з командного рядка системи MatLab або з інших m-файлів, вказавши всі необхідні атрибути – вхідні та вихідні аргументи. При виклику m–функцій в тілі іншої m–функції система допускає два *формати виклику*: формат функції та формат команди.

Виклик у форматі функції майже не відрізняється від виклику функцій в інших мовах програмування. Відмінність полягає лише в тому, що система MatLab допускає наявність більше одного аргументу в списку вихідних змінних. В цьому випадку всі вихідні аргументи розміщуються в квадратних дужках й відокремлюються один від одного комами, наприклад:

[outl, out2,...,outN] = function_name(in1, in2, ..., inN)

Виклик у форматі команди формується таким чином: *function_name in*1 ... *inN*, де *in1*, ..., *inN* – вхідні аргументи. Такий формат не дозволяє сформувати вихідний аргумент.

Під час виклику m-функції їй необхідно передати дані для обробки, використовуючи вхідні аргументи. В системі існує таке правило: якщо виклик реалізується в форматі функції, то вхідний аргумент передається своїм значенням, а якщо в форматі команди, то вхідний аргумент передається своїм ім'ям у вигляді рядка символів. Наприклад, наступні дві форми виклику функції *abs* є рівносильними для A=–65:

– формат команди:

```
>>abs  A
ans = 65
```

– формат функції:

```
>>abs ('A')
ans = 65
```

Формування вхідного та вихідного масивів аргументів. Система MatLab за допомогою параметрів *varargint* і *varargout* дозволяє передавати

довільну кількість вхідних і вихідних змінних. Функція, яка повертає змінну кількість вихідних значень, розпочинається з заголовку:

function [varargout] = ім'я_функції (аргументи)

У цьому випадку система розміщує всі вхідні аргументи в масив комірок з ім'ям varargin і запам'ятовує кількість аргументів, які передаються функції в глобальній змінній *nargin*. До визначеної таким чином функції можна звертатися, використовуючи різну кількість вихідних параметрів, наприклад:

[*y*1, *y*2, *y*3] = ім'я_ функції (аргументи)

[zl, z2] = ім'я_функції (аргументи)

*w*l = ім'я_функції (аргументи)

В першому випадку змінній *y*1 буде присвоєно значення комірки *varargout*(*1*), змінній *y*2 – значення комірки *varargout*(*2*), змінній *y*3 – значення комірки *varargout*(*3*). В другому випадку будуть використані значення перших двох комірок, в третьому – значення першої комірки масива *varargout*.

Наведена нижче функція *testvarin* виводить на екран лінії, які з'єднують послідовність точок, що описуються координатами x і y. Ця функція допускає в якості вхідного аргумента будь-яку кількість зазначених точок.

```
function testvarin(varargin)
% Функція TESTVARIN(varargin) будує лінії, які з'єднують
% послідовність точок, заданих векторами з координатами [x y]
for i=1:length(varargin)
 x(i)=varargin{i}(1); y(i)=varargin{i}(2);
end
xmin = min(0, min(x));ymin = min(0, min(y));
axis([xmin fix(max(x))+3 ymin fix(max(y))+3])
plot(x,y)
```

Функція *testvarin* може працювати зі списками вхідних змінних різної довжини, а наведені нижче оператори дозволяють побудувати на одному графіку зображення двох різних послідовностей точок (рис. 1.5):

```
testvarin([2 3], [1 5], [3 4], [6 5], [4 2], [2 3]), hold on
testvarin([-1 0], [3 -5], [4 2], [1 1])
```

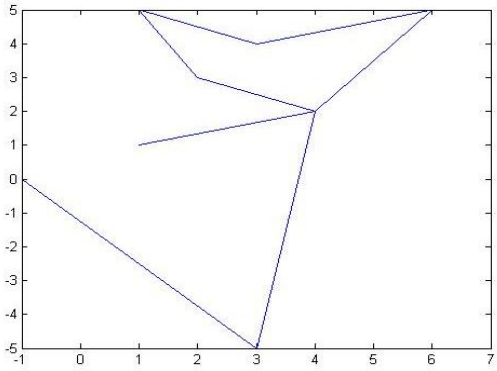


*Рис. 1.10***.** Результат виклику m-функції *testvarin*

Робоча область m-функції. При виконанні кожної m-функції для неї виділяється область пам'яті, яка називається *робочою областю m-функції.*

Під час роботи з системою MatLab можна одержати доступ лише до змінних, які розташовані в робочій області системи Workspace або в робочій області m-функції. Змінні, розташовані в робочій області m-функції, називаються *локальними***.** При завершенні роботи m-функції ці змінні не зберігаються в оперативній пам'яті.

Іноді виникає потреба використовувати деякі змінні в багатьох m-функціях. Для того, щоб відрізнити їх від локальних змінних, вони оголошуються як *глобальні змінні.* Якщо деякі змінні Workspace оголошені глобальними (за допомогою оператора *global*) і таке оголошення з вказуванням імен змінних має місце в тілі функції, то відповідна функція має доступ до вказаних змінних. Для роботи з глобальними змінними необхідно: 1) оголосити на початку програми змінну як глобальну в кожній m-функції, яка використовує цю змінну; 2) щоб змінна робочої області була глобальною, необхідно оголосити її як глобальну в головному файлі-сценарії. Опис операторів оголошення змінних, виконання m-функцій, керування процесом виконання та ведення діалогу наведено в роботі [8]. Деякі з цих операторів наведені в Додатку Б.

Тема 2. Нечіткі множини: формалізація та параметризація нечітких множин

2. Формалізація та параметризація нечітких множин

Мета роботи: ознайомитися з поняттям нечітка множина, методи її визначення (формалізація та параметризація). *Об'єкт дослідження* – нечіткі множини.

Питання для опрацювання

1. Визначення поняття «нечітка множина» [4].
2. Способи визначення нечіткої множини [4].
3. Види функцій належності, їх опис та параметризація [4, 17].

1. Задання до практичної роботи

1. Вивчити теоретичний матеріал.
2. Проаналізувати постановку задачі, дані та визначити універсум.
3. Визначити відповідні базові нечіткі множини (нечітку множину описати у вигляді структури даних) та, при необхідності, множини, отримані на основі модифікаторів.
4. Побудувати графіки нечітких множин в середовищі MatLab: з використанням та без використання внутрішніх функцій системи. Порівняти отримані графіки
5. Побудувати відповідні графіки нечітких множин в інтегрованому середовищі розробки програм на мові Python.

2. Контрольні запитання та завдання

1. Що таке «нечітка множина»? Які Ви знаєте способи визначення нечіткої множини? Опишіть нечіткі множини «істина» та «хибність».
2. Назвіть види функцій належності (ФН), опишіть їх.
3. Назвіть методи визначення ФН нечітких множин.
4. Назвіть основні характеристики нечітких множин.
5. Назвіть основні операції над нечіткими множинами.
6. Опишіть лінгвістичні модифікатори нечітких множин.
7. Визначіть поняття «нечітка» та «лінгвістична» змінні.

Задачі для самостійного розв'язання: *формалізація та параметризація нечітких множин*

1. Нехай функція належності (ФН) з двома параметрами l та r ($l<r$) є S-подібною відкритою справа ФН, яка визначається як:

$$S(x;l,r)=\begin{cases}0,\ x\le l;\\ 2\left(\dfrac{x-l}{r-l}\right)^2,\ l<x\le\dfrac{l+r}{2};\\ 1-2\left(\dfrac{r-x}{r-l}\right)^2,\dfrac{l+r}{2}<x\le r;\\ 1,r<x.\end{cases}$$

де $x\in$ [0,15]; l=1; r=9. Маємо інтервали: [0, 1]; [1, 5]; [5, 9]; [9, 15]

А. Написати функцію для реалізації цієї ФН;

Б. Побудувати графіки цієї ФН з різними значеннями параметрів;

В. Визчити точку кросовера S($x; l, r$);

Г. Навести приклади S-подібних ФН.

2. Нехай ФН з двома параметрами l та r ($l<r$) є Z-подібною ФН, яка визначається як:

$$Z(x;l,r)=1-S(x;l,r),\ \text{де}\ \ S(x;l,r)=\begin{cases}0,x\le l;\\ 2\left(\dfrac{x-l}{r-l}\right)^2,\ l<x\le\dfrac{l+r}{2};\\ 1-2\left(\dfrac{r-x}{r-l}\right)^2,\ \dfrac{l+r}{2}<x\le r;\\ 1,r<x.\end{cases}$$

де $x\in$ [0,15]; l=1; r=9. Маємо інтервали: [0, 1]; [1,5]; [5,9]; [9,15]

А. Написати функцію для реалізації цієї ФН;

Б. Побудувати графіки цієї ФН з різними значеннями параметрів;

В. Визчити точку перетину (кросовера) Z (x; l, r);

Г. Навести приклади Z-подібних ФН.

3. Нехай π-подібна ФН з двома параметрами a, c визначається через S- і Z-ФН, а саме:

$$\pi(x;a,c)=\begin{cases}S(x;c-a,c),x\le c;\\ Z(x;c,c+a),x>c;\end{cases}\ \text{де}\ \ S(x;l,r)=\begin{cases}0,x\le l;\\ 2\left(\dfrac{x-l}{r-l}\right)^2,l<x\le\dfrac{l+r}{2};\\ 1-2\left(\dfrac{r-x}{r-l}\right)^2,\dfrac{l+r}{2}<x\le r;\\ 1,r<x;\end{cases}$$

$$Z(x;l,r)=1-S(x;l,r)$$

Тут c – центр, a>0 – поширення по обидві сторони від ФН, $x\in$ [0,20]; l=1; r=9.

А. Написати функцію для реалізації цієї ФН.

Б. Побудувати графіки цієї ФН з різними значеннями параметрів;

В. Визначити точки перетину і ширину ФН π(x; l, r).

Г. Навести приклади π-подібних ФН.

4. Двостороння π-подібна ФН є продовженням π-подібна ФН, введеної у впр. 3. Вона визначається чотирма параметрами a, b, c, d:

$$ts\_\pi(x;a,b,c,d)=\begin{cases}0, x\le a;\\ S(x;a,b),\ a<x\le b;\\ 1,\ b<x\le c;\\ Z(x;c,d), c<x\le d;\\ 0, d\le x.\end{cases}$$

А. Написати функцію для реалізації цієї ФН.

Б. Побудувати графіки цієї ФН з різними значеннями параметрів;

В. Визначити точки перетину і ширину ФН ts-π(x; a, b,c,d).

Г. Навести приклади даної ФН.

5. Двостороння ФН Гаусса визначається як:

$$ts\_gaussian(x;c_1,\sigma_1,c_2,\sigma_2)=\begin{cases}exp\left[-\frac{1}{2}\left(\frac{x-c_1}{\sigma_1}\right)^2\right],\ x\le c_1;\\ 1,\ c_1<x<c_2;\\ exp\left[-\frac{1}{2}\left(\frac{x-c_2}{\sigma_2}\right)^2\right],\ c_2\le x.\end{cases}$$

А. Написати функцію для реалізації цієї ФН.

Б. Побудувати графіки цієї ФН з різними значеннями параметрів;

В. Визначити точки перетину і ширину ФН ts–π (x; a, b,c,d).

Г. Навести приклади даної ФН.

6. (аналог прикл. 2.1) Використовуючи ФН «молодий» та «старий», визначені на універсамі X=[0, 100] параметру вік людини, побудувати графіки ФН, які відповідають лінгвістичним виразам: більш-менш старий; не молодий і не старий; молодий, але не занадто молодий; дуже старий.

7. (аналог прикл. 2.1) Використовуючи ФН «молодий» та «старий», визначені на універсамі X=[0, 100], побудувати графіки ФН, які відповідають заданим лінгвістичним виразам: не дуже молодий і не дуже старий; дуже молодий або дуже старий.

8. (аналог прикл. 2.1) Використовуючи ФН «холодно» та «жарко», визначені на універсамі U =[0, 25] для параметру температура повітря в кімнаті, побудувати графіки ФН, які відповідають лінгвістичним виразам: більш-менш жарко; не холодно і не жарко; холодно, але не занадто холодно; дуже жарко.

9. (аналог прикл. 2.1) Використовуючи ФН «жарко» та «холодно», визначені на універсамі X=[0, 25] для параметру температура повітря в кімнаті, визначити задані нечіткі множини та множини, які відповідають заданим лінгвістичним виразам: не дуже холодно і не дуже жарко; дуже холодно або дуже жарко. Побудувати графіки їх ФН.

10. (аналог прикл. 2.2) Розглянемо параметр вік людини та нечіткі множини «молодий», не дуже молодий; «середнього віку», більш-менш середнього віку; «старий», дуже старий. Визначити задані нечіткі множини, побудувати графіки їх ФН.

11. (аналог прикл. 2.2) Розглянемо параметр артеріальний тиск людини та нечіткі множини «низький», не дуже низький; «помірний», більш-менш помірний; «високий», дуже високий. Визначити задані нечіткі множини, побудувати графіки їх ФН.

12. (аналог прикл. 2.2) Розглянемо параметр систолічний тиск людини та нечіткі множини «низький», не дуже низький; «помірний», більш-менш помірний; «високий», дуже високий. Визначити задані нечіткі множини, побудувати графіки їх ФН.

13. (аналог прикл. 2.2) Розглянемо параметр рівень гемоглобіну в крові людини (універсам U=[100, 200]) та нечіткі множини:

- «низький 100–130», не дуже низький;
- «нормальний 130–170», більш-менш помірний;
- «високий 170–200», дуже високий.

Визначити задані нечіткі множини, побудувати графіки їх ФН.

14. (аналог прикл. 2.2) Розглянемо параметр рівень холестерину в крові людини та нечіткі множини «низький», не дуже низький; «помірний», більш-менш помірний; «високий», дуже високий. Визначити задані нечіткі множини, побудувати графіки їх функцій належності.

15 (аналог прикл. 2.2) Розглянемо параметр вага та нечіткі множини «мала», не дуже; «помірна», більш-менш помірна; «велика», дуже велика. Визначити задані нечіткі множини, побудувати графіки їх ФН.

16. (аналог прикл. 2.8). Розглянемо параметр зріст, який визначає таке відношення:

$$TALL = \int_x \mu_{TALL}(x)/x = \int_5^8 \mu_{TALL}(x)/x = \int_5^6 \frac{(x-5)^2}{2}/x + \int_6^7 \left(1-\frac{(x-7)^2}{2}\right)/x + \int_7^8 1/x\,.$$

Побудувати графіки відповідних ФН.

17. (аналог прикл. 2.1) Використовуючи ФН «низький» та «високий», визначені на універсумі X=[0, 200] для параметру зріст людини, побудувати графіки ФН, які відповідають лінгвістичним виразам: більш-менш високий; не низький і не високий; низький, але не занадто низький; дуже високий.

18. Розглянемо параметр високий зріст, X∈[0, 10]. Нехай задано нечіткі множини: $\mu_{\text{ВИСОКИЙ ЗРІСТ}}(5) = \overline{A}1$ – менший за середній, $\mu_{\text{ВИСОКИЙ ЗРІСТ}}(6) = \overline{A}2$ – середній, $\mu_{\text{ВИСОКИЙ ЗРІСТ}}(7) = \overline{A}3$ – вищій за середній. Побудувати графіки їх ФН, якщо математично наведені ФН описуються у вигляді: 1) $\mu_{a1}(x) = 1 - S(x; 4.5, 5.5)$; 2) $\mu_{a2}(x) = \prod(x;\ 1,\ 5.5)$; 3) $\mu_{a3}(x) = S(x; 5.5, 6.5)$

19. (аналог прикл. 2.1) Нехай X – це користувачі однорангової локальної мережі (кількість користувачів в таких мережах зазвичай не перевищує 10). Для

лінгвістичної змінної «Користувачі» змінна $x \in [1, 10]$ і є базовою змінною. Тоді терм множина лінгвістичної змінної Користувачі може включати такі значення: Т(Користувачі)={<мало>, <не дуже мало>, <не мало и не багато>, <не дуже багато>, <багато>}. Визначити складові Т(Користувачі).

20. (аналог прикл. 2.2) Нехай всі можливі оцінки на екзамені U=[10, 100]. Терм множина лінгвістичної змінної Оцінки може включати такі значення: Т(Оцінки)={<низькі оцінки>, <не дуже низькі оцінки>, «середні оцінки», <не низькі і не високі оцінки>, <не дуже високі оцінки>, <високі оцінки>}. Визначити складові Т(Оцінки).

21. (аналог прикл. 2.2) Нехай всі можливі значення температури людини U=[35, 45]. Терм множина лінгвістичної змінної Температура_людини може включати такі значення: Т(Температура_тіла_людини)={<низька>, <не дуже низька>, «середня», <не низька і не висока>, <не дуже висока>, <висока>}. Визначити складові Т(Температура_тіла_людини).

22. (аналог прикл. 2.2) Нехай всі можливі значення температури води U=[0, 100]. Терм множина лінгвістичної змінної Температура_води може включати такі значення: Т(Температура_води)={<низька>, <не дуже низька>, «середня», <не низька і не висока>, <не дуже висока>, <висока>}. Визначити складові Т(Температура_води).

23. (аналог прикл. 2.2) Нехай всі можливі значення температури повітря U=[–40, 50]. Терм множина лінгвістичної змінної Температура_повітря може включати такі значення: Т(Температура_повітря)={<низька>, <не дуже низька>, «середня», <не низька і не висока>, <не дуже висока>, <висока>}. Визначити складові Т(Температура_повітря).

Аудиторна робота

Приклад 2.1. Використовуючи ФН «молодий» і «старий», визначені на універсумі X=[0, 100] таким чином:

$\mu_{молодий}(x)$=zmf(x,[0 45]) – Z-подібна; $\mu_{старий}(x)$=smf(x,[40 100]) – S-подібна

(або $\mu_{молодий}(x) = \text{gaussmf}(x, 0, 20) = e^{-\left(\frac{x}{20}\right)^2}$; $\mu_{старий}(x) = \text{gaussmf}(x, 100, 30) = e^{-\left(\frac{x-100}{30}\right)^2}$ – π-подібна типу Гауса), описати (формалізувати) ФН, які відповідають лінгвістичним виразам: більш-менш старий; не молодий і не старий; молодий, але не занадто молодий; дуже старий (дуже дуже дуже старий) з використанням MATLAB.

Розв'язання. Наведені ФН математично описуються за формулами:

1) більш–менш старий=DIL(old)=old$^{0.5}$= $\int_X \sqrt{\frac{1}{1+\left(\frac{x-100}{30}\right)^6}} / x$;

2) не молодий і не старий=$\neg$young $\cap$ $\neg$old

$$= \int_X \left[1 - \frac{1}{1+\left(\frac{x}{20}\right)^4} \right] \wedge \left[1 - \frac{1}{1+\left(\frac{x-100}{30}\right)^6} \right] / x;$$

3) молодий, але не занадто молодий = young $\cap \neg$ young^2=

$$= \int_X \left[\frac{1}{1+\left(\frac{x}{20}\right)^4} \wedge \left(1 - \left(\frac{1}{1+\left(\frac{x}{20}\right)^4} \right)^2 \right) \right] / x;$$

4) дуже дуже дуже старий=CON(CON(CON(old)))=(((old)^2)^2)^2=

$$= \int_X \left[\frac{1}{1+\left(\frac{x-100}{30}\right)^6} \right]^8 / x.$$

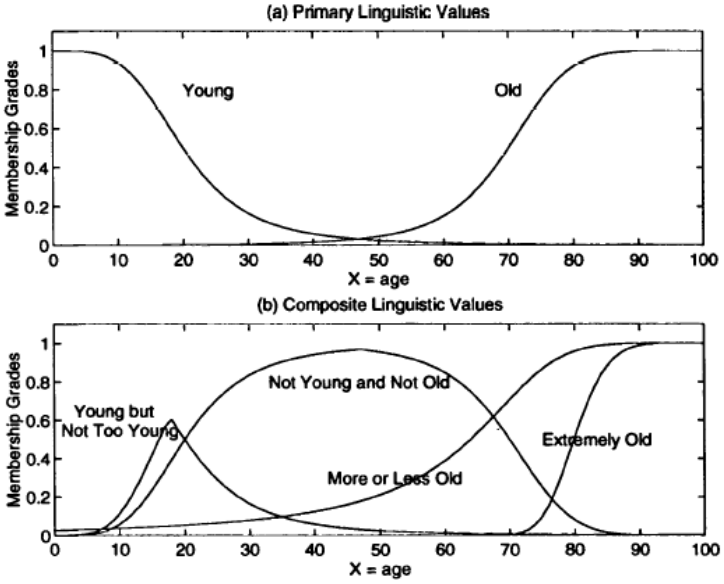


Рис. 2.1. Графіки ФН нечітких множин, які відповідають прикладу: *а* – базові ФН; *б* – модифіковані ФН

Приклад 2.2. Розглянемо параметр температура повітря в кімнаті. Описати ФН нечітких множин «холодно, не дуже холодно», «жарко (наприклад, 25°С і більше), дуже жарко», «комфортно (наприклад, 17-23°С), більш-менш комфортно».

Розв'язання. Наведені ФН математично описуються таким чином: $\forall x \in$ X=[0, 35]

1) холодно $\mu_{\bar{A}1}(x)$ (рис. 2.2): $\mu_{a1}(x) = \frac{1}{1+\left|\frac{x-1}{14}\right|^{12}}$ або цю нечітку множину описує Z-подібна функція належності сигмоїдного типу;

2) жарко $\mu_{\bar{A}3}(x)$ (рис. 1.2): $\mu_{a3}(x)=\dfrac{1}{1+\left|\dfrac{x-40}{16}\right|^{24}}$ або цю нечітку множину описує S-подібна ФН сигмоїдного типу;

3) комфортно $\mu_{\bar{A}2}(x)$ (рис. 1.2): $\bar{A}2$ – трапецієвидна функція належності вигляду: $\mu_{a2}(x;a,b,c,d)=\begin{cases}0, & x\le a;\\ \dfrac{x-a}{b-a}, & a<x\le b;\\ 1, & b<x\le c;\\ \dfrac{d-x}{d-c}, & c<x\le d;\\ 0, & d<x;\end{cases}$ якщо a=15, b=18, c=23, d=25 або цю нечітку множину описує π-подібна ФН $\mu_{a2}(x)=\dfrac{1}{1+\left|\dfrac{x-20}{3}\right|^{8}}$.

```
clear all;close all;clc; x=[0:0.5:40];
m1=((1+abs((x-1)/14).^12).^(-1));  % холодно
m2=((1+abs((x-40)/16).^24).^(-1)); % жарко
m3=((1+abs((x-20)/3).^8).^(-1));   % комфортно
m11=1-m1.^2;  % не дуже холодно
m12=m3.^(1/2);% більш-менш комфортно
m13=m2.^2;    % дуже жарко
figure(1), plot(x,m1,'k-',x,m2,'-.k',x,m3,'k--');grid on; hold on;
plot(x,m11,'-ko',x,m12,'k.',x,m13,'kx');grid on;
xlabel('Температура');
h=legend('холодно','жарко','комфортно','не дуже холодно',...
'б.-м.комфортно','дуже жарко','Location','BestOutside');
set(h,'Interpreter','onne')
```

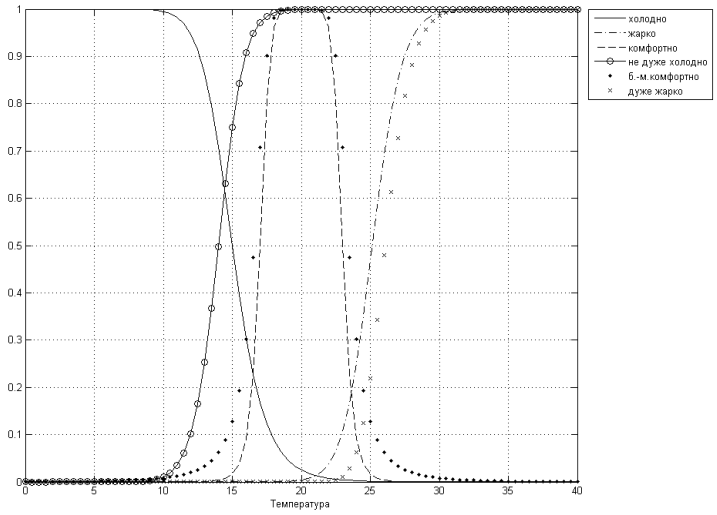


Рис. 2.2. Графіки ФН нечітких множин $\bar{A}1$, $\bar{A}2$, $\bar{A}3$, які відповідають нечітким змінним α_1, α_2, α_3 для лінгвістичної змінної
В = температура_повітря (α1= «холодно», α2 = «комфортно», α3 = «жарко»)

Приклад 2.3. Розглянемо параметр відстань до роботи. Нехай задано нечіткі множини «близько», «далеко (наприклад, 4000 м і більше)», «комфортно (наприклад, 1000 ÷ 4000 м)». Побудувати графіки їх ФН, якщо математично наведені функції належності описуються як:

1) близько $\mu_{\bar{A}1}(x)$: $\forall x \in X=[0,\ 5000]$ $\mu_{a1}(x)=\dfrac{1}{1+\left|\dfrac{x}{1000}\right|^{12}}$, або цю нечітку множину описує Z-подібна функція належності сигмоїдного типу, яку можна у загальному випадку задати у вигляді: $f_{S1}(x;a,b)=\dfrac{1}{1+e^{-a(x-b)}}$.

2) далеко $\mu_{\bar{A}3}(x)$: $\forall x \in X=[0,\ 5000]$ $\mu_{a3}(x)=\dfrac{1}{1+\left|\dfrac{x-5000}{1000}\right|^{12}}$, або цю нечітку множину описує S-подібна функція належності сигмоїдного типу, яка у загальному випадку задається у вигляді: $f_{S2}(x;a,b)=\dfrac{1}{1+e^{-a(x-b)}}$.

3) комфортно $\mu_{\bar{A}2}(x)$: $\bar{A}2$ – трапецієвидна ФН вигляду: $\forall x \in X=[0,\ 5000]$ $\mu_{a2}(x)=\dfrac{1}{1+\left|\dfrac{x-2500}{1500}\right|^{12}}$.

Алгоритм виконання роботи

Крок 1.Формуємо універсум (від 0 до 5000 м).

Крок 2. Визначаємо ФН відповідних нечітких множин.

Крок 3. Будуємо графіки та виводимо результат.

I. В середовищі MatLab. Функції належності нечітких множин «близько», «далеко», «комфортно» можна задавати у вигляді звичайних неперервних функцій, які мають відповідну до умови форму та властивості. Це дозволяє візуалізувати ФН для кращого сприйняття та подальшої обробки цих множин.

Лістинг програми 1

```
clear all;close all;clc;
x=[0:0.2:10];
near1=(1+(abs(x/2)).^4).^-1;
far1=(1+(abs((x-10)/3)).^6).^-1;
middle1=(1+(abs((x-5)/2)).^6).^-1;

near2=zmf(x,[0,4]); far2=smf(x,[6,10]); middle2=pimf(x,[0,4,5,9]);
%або middle2=gaussmf(x,[1.5,5]);

figure(1)
plot(x,near1,'k--',x,far1,'kx',x,middle1,'k.');grid on;
hold on;
```

```
xlabel('Distance');legend('Близько','Далеко','Нормально')
figure(2)
plot(x,near2,'k--',x,far2,'kx',x,middle2,'k.');grid on
hold on;xlabel('Distance');
```

Приклад використання внутрішніх π-функцій – *pimf*, *trapmf*

```
x = -10:0.1:10;
mf = trapmf (x,[-10 -8 -4 7]); plot(x, mf, '-')
```

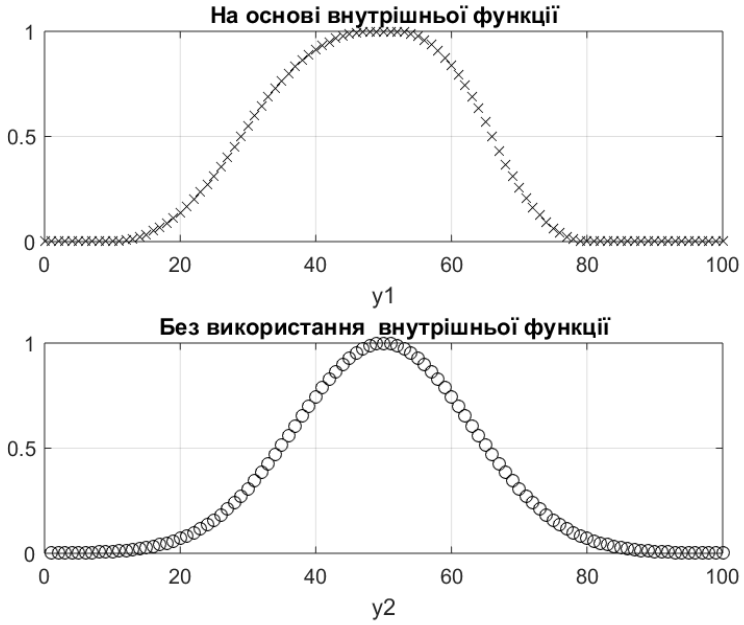
Функція *pimf*

```
pimf(x,[a,b,c,d])=smf(x,[a,b]))*zmf(x,[c,d]))
```

Лістинг програми 2

```
clc; clear all; close all;
x=0:100
a=10;b=48;c=52;d=80
figure(1); y1=pimf(x,[a,b,c,d]);
subplot(2,1,1),plot(x,y1, 'kx'), grid on;
xlabel('y1'); title('На основі внутрішньої функції')
y2=1:100
for i=1:100
 y2(i)=exp(-(i-50)^2/(2*13^2))
end;
subplot(2,1,2), plot(1:100,y2, 'ko'), grid on; xlabel('y2');
title('Без використання внутрішньої функції')
```

Результат



Приклад 2.4 (аналог прикл. 2.1) Розглянемо параметр «вік людини». Нехай задано нечіткі множини «молодий», «старий». Визначити ці нечіткі множини та деякі їхні модифікації, побудувати відповідні графіки функцій належності.

Лістинг програми 1:

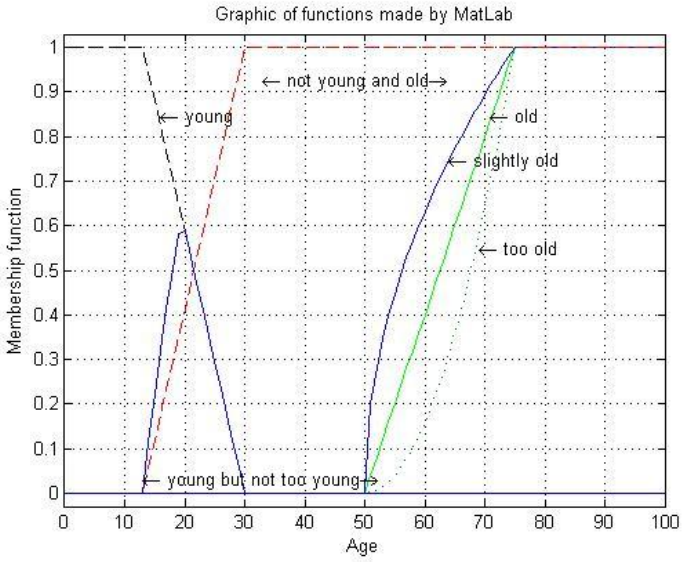
```
function main
     clc;close all; clear all;x=0:1:100;
     figure(1);c=0;sig=20;t1=f1(x,c,sig);
     figure(1);subplot (2,1,1), plot(x,t1,'r'), grid on;
     xlabel ('age'), title ('Молодий');
     c=100;sig=30;t2=f1(x,c,sig);
     subplot (2,1,2), plot(x,t2,'r'), grid on;
     xlabel ('age'), title ('Старий');
     figure(2);t=t2.^(1/2);
     subplot (5,1,1), plot(x,t,'r'), grid on;
     xlabel ('age'), title ('Більш-менш старий');
     subplot (5,1,5), plot(x,t), hold on;
     t=min((1-t1),(1-t2));
     subplot (5,1,2), plot(x,t,'g'), grid on;
     xlabel ('age'), title ('Не молодий і не старий');
     subplot (5,1,5), plot(x,t), hold on;

function y=f1(a,b,d);
     y=exp((-1/2)*((a-b).^2)/d^2);
```

Лістинг програми 2:

```
function main
clc; clear all;close all;
x=0:1:100;
a1=0;b1=0;c1=13;d1=30;  a2=50;b2=75;c2=100;d2=100;
% Розрахунок ФН
young=trapmf (x,[a1 b1 c1 d1]);
old=trapmf (x,[a2 b2 c2 d2]);
[l,k,j,h]=m_for_MatLab(old, young);
mlo=l; too_old=k; nor=j; y_but_not_too_y=h;

youngs=struct  ('x',  x,  'y',  young,  'y_but',  y_but_not_too_y,
'nory', nor);
olds=struct ('x', x, 'o', old, 't', too_old, 'ml', mlo);
% ------------Побудова графіка-------------
figure(1)
plot  (youngs.x,  youngs.y,  'k--',  olds.x,  olds.o,  'g-',  olds.x,
olds.ml,olds.x,   olds.t,   ':',   youngs.x,   youngs.nory,   'r--',
youngs.x, youngs.y_but, 'b-');grid on;
ylim  ([-0.03,  1.03]);  title  ('Graphic  of  functions  made  by
MatLab');
ylabel('Membership function'); xlabel ('Age');
text (71, 0.85, '\leftarrow old');
text (16, 0.85, '\leftarrow young');
text (64, 0.75, '\leftarrow slightly old');
text (69, 0.55, '\leftarrow too old');
text (33, 0.93, '\leftarrow not young and old\rightarrow');
text (13, 0.035, '\leftarrow young but not too young\rightarrow');
% Розрахунок ФН
[o, y]= membership (x,a1,b1,c1,d1,a2, b2, c2, d2);
```

```
old=o; young=y;
[l,k,j,h]=m_for_MatLab (old, young);
mlo=l; too_old=k; nor=j; y_but_not_too_y=h;

%youngs=struct ('x', x, 'y', young, 'y_but', y_but_not_too_y,
'nory', nor);
%olds=struct ('x', x, 'o', old, 't', too_old, 'ml', mlo);
figure(2)
plot (youngs.x, youngs.y, 'k--', olds.x, olds.o, 'g-', olds.x,
olds.ml, olds.x,olds.t, ':', youngs.x, youngs.nory, 'r--',
youngs.x, youngs.y_but, 'b-'); grid on; ylim ([-0.03, 1.03]);
title ('Graphic of functions made by me');
ylabel('Membership function'); xlabel ('Age');
text (71, 0.85, '\leftarrow old');
text (16, 0.85, '\leftarrow young');
text (64, 0.75, '\leftarrow slightly old');
text (69, 0.55, '\leftarrow too old');
text (33, 0.93, '\leftarrow not young and old\rightarrow');
text (13, 0.035, '\leftarrow young but not too young\rightarrow');
figure(3)
plot (youngs.x, youngs.y, 'k--', olds.x, olds.o, 'g-', olds.x,
olds.ml,olds.x, olds.t, ':', ...
youngs.x, youngs.nory, 'r--', youngs.x, youngs.y_but, 'b-');
hold on;
plot(x, young, 'k--', x,old, 'g-', x, mlo,x, too_old,':',x, nor,
'--', x, y_but_not_too_y,'b-'); grid on;
ylim ([-0.03, 1.03]);
title ('Compare functions');
ylabel('Membership function'); xlabel ('Age');
text (71, 0.85, '\leftarrow old');
text (16, 0.85, '\leftarrow young');
text (64, 0.75, '\leftarrow slightly old');
text (69, 0.55, '\leftarrow too old');
text (33, 0.93, '\leftarrow not young and not old\rightarrow');
text (13, 0.035, '\leftarrow young but not too young\rightarrow');

function [mlo,too_old,nor,y_but_not_too_y]=m_for_MatLab(old, young)
mlo=old.^0.5; too_old=old.^2;
too_young=young.^2; not_young=zeros(length(young));
not_old=zeros(length(old)); nor=not_young;
not_too_young=not_young; y_but_not_too_y=not_young;

for i=1:1:101
 not_young(i)=1-young (i); not_old(i)=1-old(i);
 not_too_young(i)=1-too_young(i);
 nor(i)=min(not_young(i), not_too_young(i));
 y_but_not_too_y(i)=min(young(i), not_too_young(i));
end
```

Результат: графіки ФН нечітких множин: «молодий», «старий» та їх модифікацій



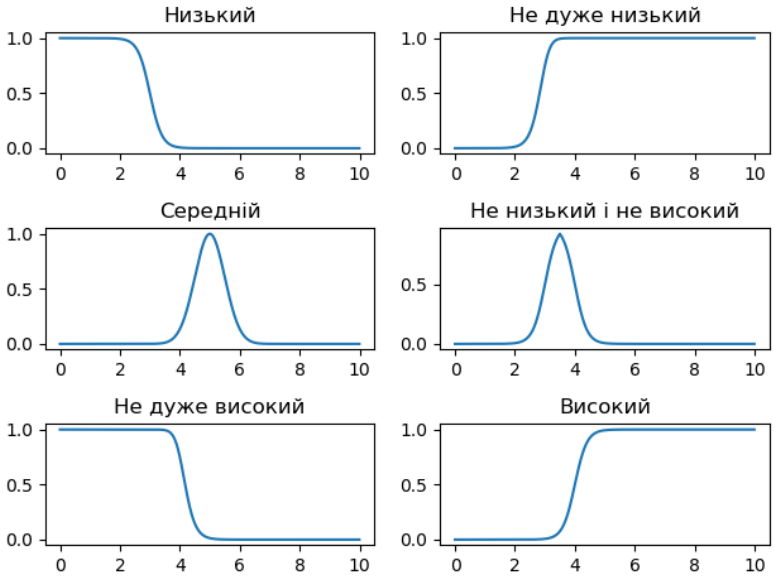
II. В Інтегрованому середовищі розробки програм на мові Python

Приклад 2.5. Побудувати графіки ФН: {низький, середній, високий} параметру «рівень гемоглобіну в крові» та їхні модифікації.

Лістинг програми 1

```
import numpy as np
import matplotlib.pyplot as plt

def sFunc(x,a,b):
 return (1+np.exp(-a*(x-b)))**(-1)
def pFunc(x,s,c):
 return np.exp((-(x-c)**2)/(2*s**2))

x=np.arange(0,10,0.01)
f,plots=plt.subplots(3,2); low=sFunc(x-1,-5,2);
plots[0,0].plot(x,low); plots[0,0].set_title('Низький(low)')
notTooLow=1-low**2;
plots[0,1].plot(x,notTooLow);
plots[0,1].set_title('Не дуже низький)')
medium=pFunc(x-4,0.5,1);
plots[1,0].plot(x,medium);plots[1,0].set_title('Середній)')

high=sFunc(x-3,5,1);
plots[2,1].plot(x,high);plots[2,1].set_title('Високий)')

u=1-high**2; plots[2,0].plot(x,u)
plots[2,0].set_title('Не дуже високий)')
temp1=1-low;temp2=1-high
for i in range(temp1.size):
  u[i]=min(temp1[i],temp2[i])
plots[1,1].plot(x,u);
plots[1,1].set_title('Не низький і не високий)')
plt.tight_layout(); plt.show();
```

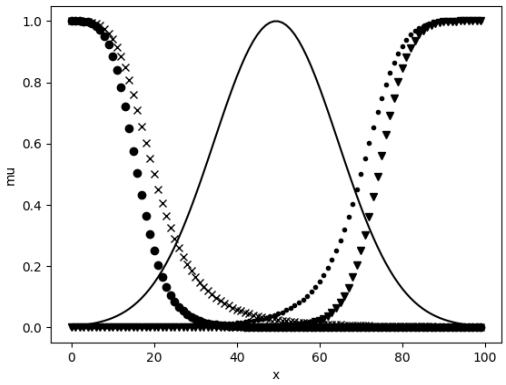
Результат



Лістинг програми 2

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

def sFunc(t):
 return np.power((1+ np.power((t/20),4)),-1)
def zFunc(t):
 return np.power((1+np.power(((t-100)/30),6)),-1)
x=np.arange(0,100,1); gauss=signal.gaussian(100, std=15)
y=sFunc(x)**2; y1=zFunc(x)**2
plt.plot(x,sFunc(x),'kx',x,zFunc(x),'k.',x,gauss,'k',x,y,'ko',x,
y1,'kv')
plt.xlabel('x'); plt.ylabel('mu')
plt.show();
```

Результат



Приклад 2.6.

1. Побудувати графіки ФН {молодий, старий} з використанням функцій бібліотеки *skfuzzy*

Лістинг програми

```
import skfuzzy as fuzz
```

```
import matplotlib.pyplot as plt
import numpy as np
x=np.arange(0,100,1)
fig, (ax0, ax1)= plt.subplots(nrows=2, figsize=(8,9))
# базові ФН
t1=fuzz.zmf(x,0,20); t2=fuzz.smf(x,70,100)
# t1=np.exp(-(x/20)**2); t2= np.exp(-((x-100)/30)**2);

ax0.plot(t1, label='молодий'); ax0.plot(t2, label='старий')
ax0.legend(); ax0.grid();
# модифіковані
ax1.plot(np.minimum((1-t1**2),(1-t2**2)), label='не дуже молодий і
не дуже старий')
ax1.plot(np.maximum(t1**2,  t2**2),label='дуже молодий або дуже
старий')
ax1.legend(); ax1.grid(); plt.xlabel('Age'); plt.show();
```

2. Побудувати графіки ФН {молодий, старий, середнього_віку} з використанням функцій бібліотеки *skfuzzy*

```
import skfuzzy
import matplotlib.pyplot as plt
import numpy as np
x=np.arange(0,100,1)
# базові ФН
plt.plot(skfuzzy.zmf(x,0,20))
plt.plot(skfuzzy.gaussmf(x,30,10))
plt.plot(skfuzzy.smf(x,35,80))
# модифіковані
plt.plot(skfuzzy.smf(x,35,80)**1/2)
plt.plot(1-skfuzzy.gaussmf(x,30,10)**2)
plt.grid(); plt.xlabel('Age'); plt.show();
```

Приклад 2.7. Нехай задана нечітка множина вигляду:

$$\overline{A}=\int_x \mu_{\overline{A}}(x)/x=\int_{180}^{210}\mu_{\overline{A}}(x)/x=\int_{180}^{190}\frac{(x-180)^2}{2}/x+\int_{190}^{200}\left(1-\frac{(x-200)^2}{2}\right)/x+\int_{200}^{210}1/x.$$

Необхідно побудувати її графік належності.

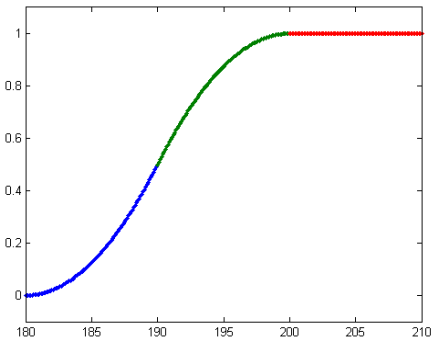
Розв'язання. **1. MatLab**

```
close all; clear all; clc;
X1=180:0.1:190-0.1; short=((X1-180).^2)/200;
X2=190:0.1:200-0.1; medium=1-((X2-200).^2)/200;
X3=200:0.1:210; tall=ones(size(X3));
plot(X1, short, '.', X2, medium, '.', X3, tall,'.')
ylim([-0.1 1.1])
```

2. Python

```
import matplotlib.pyplot as mlp
def calc(i,s1,s2,s3,s4):
```

```
  if i<s1:
   return (0)
  elif i<=s2:
   return (((i-180)*(i-180))/200)
  elif i<=s3:
   return (1-((i-200)*(i-200))/200)
  else:
   return (1)
if __name__== '__main__':
 x=[179+i/10 for i in range(0,310)]; a=len(x)
 for i in range(1,a):
   res=[calc(x[i],180,190,200,210) for i in range(0,a)]
 mlp.plot(x, res, linestyle='-'); mlp.show()
```



Теоретичні відомості

1. Визначення поняття «нечітка множина» [4]. *Універсум* – це множина, яка містить всі можливі елементи в рамках контексту визначеної предметної області. Універсум, зазвичай, позначають літерою *X*.

Нечіткою множиною (НМ) $\overline{A}$ називається множина впорядкованих пар або кортежів вигляду $< x;\ \mu_{\overline{A}}(x) >$, де x – елемент універсуму X, $\mu_{\overline{A}}(x): X \to [0,\ 1]$ – функція належності (ФН), яка ставить у відповідність кожному елементу $x \in X$ дійсне число з інтервалу [0, 1], що характеризує ступінь належності елемента x до нечіткої множини $\overline{A}$. Чим більшим є значення ФН $\mu_{\overline{A}}(x)$, тим більше елемент універсальної множини x відповідає властивостям нечіткої множини $\overline{A}$. Скінченну нечітку множину $\overline{A}$ будемо записувати у вигляді $\overline{A}=\{<x_1;\ \mu_{\overline{A}}(x_1)>, ..., <x_n;\ \mu_{\overline{A}}(x_n)>\}$, або $\overline{A}=\{<x;\ \mu_{\overline{A}}(x)>\}$, де n – кількість елементів нечіткої множини $\overline{A}$. Крім цих позначень використовують такі форми запису:

1) $\bar{A}=\{(\mu_{\bar{A}}(x_1); x_1),...,(\mu_{\bar{A}}(x_n); x_n)\}$; 2) $\bar{A}=\frac{\mu_{\bar{A}}(x_1)}{x_1}+...+\frac{\mu_{\bar{A}}(x_n)}{x_n}=\sum_{i=1}^{n}\frac{\mu_{\bar{A}}(x_i)}{x_i}$.

При цьому горизонтальна (або похила) лінії є розділовими символами, а символ «+» позначає теоретико–множинне об'єднання окремих елементів.

Неперервні нечіткі множини, як правило, записують у вигляді: $\bar{A}=\int_X \frac{\mu_{\bar{A}}(x)}{x}$, тобто нечітка множина $\bar{A}$ зображує собою об’єднання континіуму пар $\left(\frac{\mu_{\bar{A}}(x)}{x}\right)$, де «континіум» – множина дійсних чисел; інтеграл зображує об’єднання нечітких одноелементних множин $\frac{\mu_{\bar{A}}(x_i)}{x_i}$.

Порожню нечітку множину позначають символом $\varnothing$, вона визначає нечітку множину, ФН якої тотожньо дорівнює нулю ($\mu_{\varnothing}\equiv 0$). *Універсальну* НМ позначають символом *U*, вона визначає нечітку множину, ФН якої тотожньо дорівнює одиниці ($\mu_U(x) = 1$).

2. Способи визначення нечіткої множини [4]. Нечіткі множини задаються такими трьома способами:

1. *У формі списку* (або таблиці) з переліком всіх елементів нечіткої множини та відповідних їм значень ФН: $\bar{A}=\{<x_1; \mu_{\bar{A}}(x_1))>, ..., <x_n; \mu_{\bar{A}}(x_n)>\}$. При цьому елементи з нульовими значеннями ФН, зазвичай, не вказуються в даному списку. Цей спосіб застосовують для визначення нечітких множин, які мають скінченний дискретний носій.

2. *Аналітично* – у формі математичного виразу для відповідної ФН: $\bar{A}=\{<x; \mu_{\bar{A}}(x)>\}$ або $\bar{A}=\{x; \mu_{\bar{A}}(x)\}$. Цей спосіб використовують для визначення нечітких множин як зі скінченним, так і з нескінченним носієм.

3. *Графічно* – у формі деякої кривої або сукупності окремих точок у двовимірному просторі. При цьому одна координата відповідає елементам універсуму *X*, а друга – значенням ФН цих елементів.

3. Види ФН, їх опис та параметризація [4, 17]. Відомо, що НМ повністю характеризується її функцією належності. Більшість НМ використовують універсум X, який складається з підінтервалів множини $\Re$, тому непрактично перерахувати всі пари $< x;\ \mu_{\tilde{A}}(x) >$. Більш зручний спосіб визначення нечіткої множини – це зобразити її ФН математично. Визначимо декілька класів параметризованих одновимірних ФН, тобто ФН з одним входом.

Трикутна ФН задається трьома параметрами $\{a, b, c\}$, де $a < b < c$, таким чином: $\mu(x) = triangle(x; a, b, c) = \begin{cases} \frac{x-a}{b-a},\ a \le x \le b; \\ \frac{c-x}{c-b},\ b \le x \le c; \\ 0,\ x \le a \text{ або } c \le x. \end{cases}$ Якщо $(b–a) = (c–b)$ маємо симетричну трикутну ФН, яка однозначно задається двома параметрами з трійки (a, b, c).

Трапецієподібна ФН задається чотирма параметрами $\{a, b, c, d\}\}$, де $a < b \le c < d$, таким чином: $\mu(x) = trapezoid(x; a, b, c, d) = \begin{cases} \frac{x-a}{b-a},\ a \le x \le b; \\ 1,\ b \le x \le c; \\ \frac{d-x}{d-c},\ c \le x \le d; \\ 0,\ d \le x \text{ або } x \le a. \end{cases}$ Якщо $b = c$ ця ФН з параметрами $\{a, b, c, d\}$ зводиться до трикутної. Якщо $(b–a) = (d–c)$ ця ФН має симетричний вигляд.

Функція належності Гаусса (рис. 2.3) зазвичай описується формулою $\mu(x) = gaussian\ (x; c, \sigma) = e^{-\frac{(x-c)^2}{\sigma^2}}$ і визначається параметрами (c, σ), де c задає модальне значення НМ, σ – ширину. На рівні $\mu(x) = e^{-1} \approx 0.36788$ ширина функції дорівнює 2σ.

Якщо модальне значення функції Гаусса c = 170 см, то можна обчислити значення σ:

$$\mu(x) = e^{-\frac{(x\ -\ 170)^2}{\sigma^2}} = 0,5;$$

$$\sigma = \frac{|x-c|}{\sqrt{\ln 2}} \cong 6 \text{ см}.$$

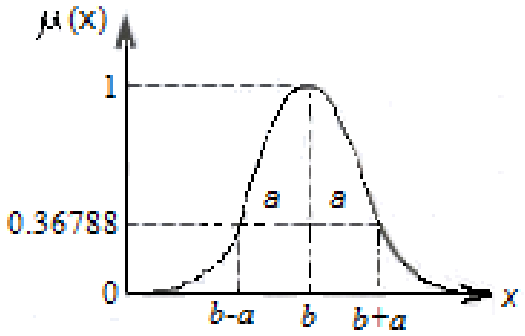
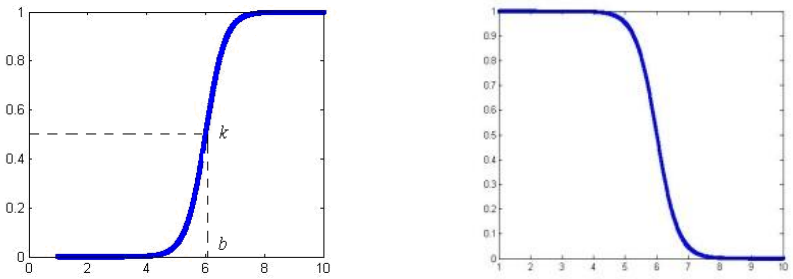


Рис. 2.3. Функція Гаусса (c=a, σ =a)

Сигмоїдна ФН визначається таким чином: $\mu(x) = sig(x;a,b) = \frac{1}{1+e^{-a(x-b)}}$, де a, b – деякі числові параметри, такі, що $a < b$. Параметр a керує нахилом в точці кросовера k: $x = b$. Залежно від знаку параметра a сигмоїдні ФН є *S*– або *Z*-подібними (за виглядом кривих, які зображують їхні графіки) і є корисними для зображення таких множин, як «дуже великий» або «дуже малий». При цьому, якщо $a > 0$, то маємо *S*–подібну ФН, інакше *Z*–подібну ФН (рис. 2.4). При цьому, якщо a>0, маємо *S*–подібну ФН, інакше *Z*–подібну.



а *б*

Рис. 2.4. Графіки сигмоїдної ФН якщо: *а*) a = 3, b = 6; *б*) a = -3, b = 6

S-функція (рис. 2.5) визначається таким чином:

$$S(x;\alpha,\beta,\gamma) = \begin{cases} 0, \text{ для } x \le \alpha; \\ 2\left(\frac{x-\alpha}{\gamma-\alpha}\right)^2, \text{для } \alpha \le x \le \beta; \\ 1-2\left(\frac{x-\gamma}{\gamma-\alpha}\right)^2, \text{для } \beta \le x \le \gamma; \\ 1, \text{ для } x \ge \gamma. \end{cases}$$

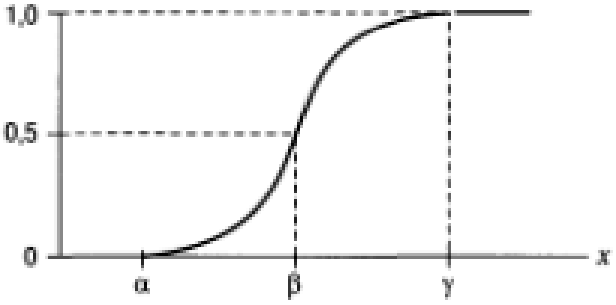


Рис. 2.5. S-подібна ФН

На рис. 2.6 зображена П-подібна ФН:

$$\Pi(x;\ \beta,\ \gamma) = \begin{cases} S(x;\gamma-\beta,\gamma-\beta/2,\gamma), & \text{якщо } x \le \gamma \\ 1-S(x;\gamma,\gamma+\beta/2,\gamma+\beta), & \text{в інших випадках} \end{cases}$$

П-функція досягає нуля в точках $x = \beta \pm \gamma$ (параметр β визначає загальну ширину), а координати точок перетину ФН з прямою $\mu = 0{,}5$ визначаються як $x = \beta \pm \gamma / 2$.

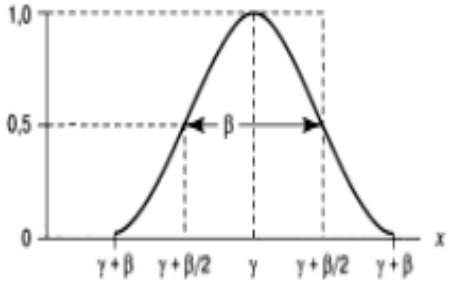


Рис. 2.6. Графік П-подібної ФН

Розглянуті види ФН утворюють список, який не є вичерпний, для конкретних програм можуть бути створені інші спеціалізовані ФН [14]. Зокрема, можна використати будь–які неперервні функції розподілу ймовірностей, за умови, що набір параметрів відповідає значенням ФН.

Розділ 2. Нечіткі моделі на основі нечіткої логіки

Тема 1. Алгоритми нечіткого виведення

3. Алгоритми нечіткого виведення. Процес розробки системи нечіткої логіки Мамдані в інтерактивному режимі

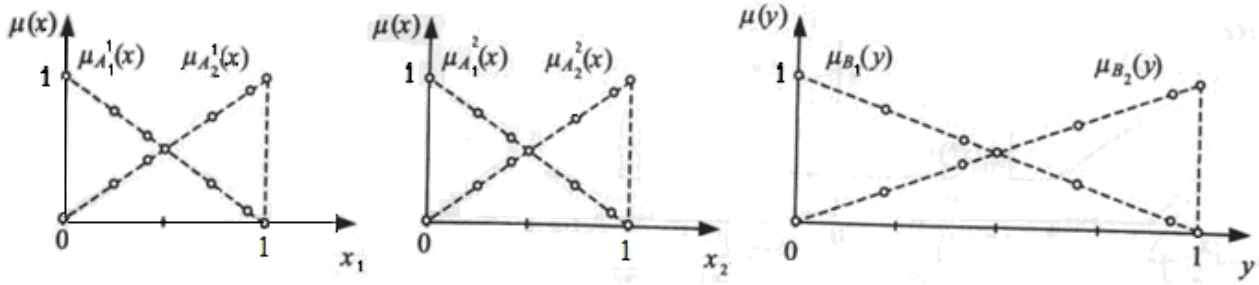
Мета роботи: ознайомитися з 1) основними алгоритмами нечіткого виведення; 2) основними поняттями та положеннями теорії нечіткого моделювання. *Об'єкт дослідження* – алгоритми нечіткого виведення Мамдані, Ларсена, Сугено.

Питання для опрацювання

1. Алгоритми нечіткого виведення Мамдані, Ларсена, Сугено [2].
2. Процес розробки СНВ Мамдані в інтерактивному режимі [3].
3. Структура FIS

1. Завдання до практичної роботи

1. Вивчити теоретичний матеріал.
2. Виконати індивідуальне завдання.
3. Сформувати відповідну базу правил (на рис. зображені функції належності нечітких множин, які використовуються в цих правилах).



Засвоїти процес розробки СНВ Мамдані в інтерактивному режимі.

4. Написати програму, яка реалізує відповідну нечітку модель: остання апроксимує аналітичну залежність, задану множиною точок, у визначеному діапазоні змінних (заняття 4, 7).

2. Контрольні запитання та завдання

1. Опишіть алгоритми нечіткого виведення Мамдані, Ларсена, Сугено та наведіть приклади.
2. Виконайте порівняльний аналіз алгоритмів Ларсена та Мамдани.

Задачі для самостійного розв'язання (варіанти завдань): *алгоритми нечіткого виведення Мамдані, Ларсена і Сугено*

Функцію $f(\mathbf{x})$ задано множиною точок у вигляді пар «вхід – значення функції». Реалізувати заданий алгоритм нечіткого виведення в точці $\mathbf{x}^*=[0{,}2;\ 0{,}8]^T$.

1. Маємо таку множину пар «вхід – значення функції»:

$\left\{\mathbf{p}_1=\begin{bmatrix}0\\0\end{bmatrix},\ t_1=0\right\},\ \left\{\mathbf{p}_2=\begin{bmatrix}0\\1\end{bmatrix},\ t_2=0\right\},\ \left\{\mathbf{p}_3=\begin{bmatrix}1\\0\end{bmatrix},\ t_3=0\right\},\ \left\{\mathbf{p}_4=\begin{bmatrix}1\\1\end{bmatrix},\ t_4=1\right\}.$

Реалізувати алгоритм Ларсена для цієї функції.

2. Завдання 1 для алгоритму нечіткого виведення Сугено.

3. Маємо таку множину пар «вхід – значення функції»:

$\left\{\mathbf{p}_1=\begin{bmatrix}0\\0\end{bmatrix},\ t_1=0\right\},\ \left\{\mathbf{p}_2=\begin{bmatrix}0\\1\end{bmatrix},\ t_2=1\right\},\ \left\{\mathbf{p}_3=\begin{bmatrix}1\\0\end{bmatrix},\ t_3=1\right\},\ \left\{\mathbf{p}_4=\begin{bmatrix}1\\1\end{bmatrix},\ t_4=1\right\}.$

Реалізувати алгоритм Мамдані для цієї функції.

4. Завдання 3 для алгоритму нечіткого виведення Ларсена.

5. Завдання 3 для алгоритму нечіткого виведення Сугено.

6. Розглянемо апроксимацію логічної функції «XOR» (*виключна диз'юнкція*). Маємо таку множину пар «вхід – значення функції»:

$\left\{\mathbf{p}_1=\begin{bmatrix}0\\0\end{bmatrix},\ t_1=0\right\},\ \left\{\mathbf{p}_2=\begin{bmatrix}0\\1\end{bmatrix},\ t_2=1\right\},\ \left\{\mathbf{p}_3=\begin{bmatrix}1\\0\end{bmatrix},\ t_3=1\right\},\ \left\{\mathbf{p}_4=\begin{bmatrix}1\\1\end{bmatrix},\ t_4=0\right\}.$

Реалізувати алгоритм Мамдані для цієї функції.

7. Завдання 6 для алгоритму нечіткого виведення Ларсена.

8. Завдання 6 для алгоритму нечіткого виведення Сугено.

9. Маємо таку множину пар «вхід – значення функції»:

$\left\{\mathbf{p}_1=\begin{bmatrix}0\\0\end{bmatrix},\ t_1=0\right\},\ \left\{\mathbf{p}_2=\begin{bmatrix}0\\1\end{bmatrix},\ t_2=0\right\},\ \left\{\mathbf{p}_3=\begin{bmatrix}1\\0\end{bmatrix},\ t_3=1\right\},\ \left\{\mathbf{p}_4=\begin{bmatrix}1\\1\end{bmatrix},\ t_4=0\right\}.$

Реалізувати алгоритм Мамдані для цієї функції.

10. Завдання 9 для алгоритму нечіткого виведення Ларсена.

11. Завдання 9 для алгоритму нечіткого виведення Сугено.

12. Маємо таку множину пар «вхід – значення функції»:

$\left\{\mathbf{p}_1=\begin{bmatrix}0\\0\end{bmatrix},\ t_1=1\right\},\ \left\{\mathbf{p}_2=\begin{bmatrix}0\\1\end{bmatrix},\ t_2=0\right\},\ \left\{\mathbf{p}_3=\begin{bmatrix}1\\0\end{bmatrix},\ t_3=0\right\},\ \left\{\mathbf{p}_4=\begin{bmatrix}1\\1\end{bmatrix},\ t_4=1\right\}.$

Реалізувати алгоритм Мамдані для цієї функції.

13. Завдання 12 для алгоритму нечіткого виведення Ларсена.

14. Завдання 12 для алгоритму нечіткого виведення Сугено.

15. Маємо таку множину пар «вхід – значення функції»:

$\left\{\mathbf{p}_1=\begin{bmatrix}0\\0\end{bmatrix},\ t_1=1\right\},\ \left\{\mathbf{p}_2=\begin{bmatrix}0\\1\end{bmatrix},\ t_2=1\right\},\ \left\{\mathbf{p}_3=\begin{bmatrix}1\\0\end{bmatrix},\ t_3=1\right\},\ \left\{\mathbf{p}_4=\begin{bmatrix}1\\1\end{bmatrix},\ t_4=0\right\}.$

Реалізувати алгоритм Мамдані для цієї функції.

16. Завдання 15 для алгоритму нечіткого виведення Ларсена.

17. Завдання 15 для алгоритму нечіткого виведення Сугено.

18. Маємо таку множину пар «вхід – значення функції»:

$\left\{\mathbf{p}_1=\begin{bmatrix}0\\0\end{bmatrix},\ t_1=1\right\},\ \left\{\mathbf{p}_2=\begin{bmatrix}0\\1\end{bmatrix},\ t_2=0\right\},\ \left\{\mathbf{p}_3=\begin{bmatrix}1\\0\end{bmatrix},\ t_3=1\right\},\ \left\{\mathbf{p}_4=\begin{bmatrix}1\\1\end{bmatrix},\ t_4=1\right\}.$

Реалізувати алгоритм Мамдані для цієї функції.

19. Завдання 18 для алгоритму нечіткого виведення Ларсена.

20. Завдання 18 для алгоритму нечіткого виведення Сугено.

21 Маємо таку множину пар «вхід – значення функції»:

$\left\{\mathbf{p}_1=\begin{bmatrix}0\\0\end{bmatrix},\ t_1=1\right\},\ \left\{\mathbf{p}_2=\begin{bmatrix}0\\1\end{bmatrix},\ t_2=0\right\},\ \left\{\mathbf{p}_3=\begin{bmatrix}1\\0\end{bmatrix},\ t_3=0\right\},\ \left\{\mathbf{p}_4=\begin{bmatrix}1\\1\end{bmatrix},\ t_4=0\right\}.$

Реалізувати алгоритм Мамдані для цієї функції.

22. Завдання 21 для алгоритму нечіткого виведення Ларсена.

23. Завдання 21 для алгоритму нечіткого виведення Сугено.

24. Маємо таку множину пар «вхід – значення функції»:

$\left\{\mathbf{p}_1=\begin{bmatrix}0\\0\end{bmatrix},\ t_1=0\right\},\ \left\{\mathbf{p}_2=\begin{bmatrix}0\\1\end{bmatrix},\ t_2=1\right\},\ \left\{\mathbf{p}_3=\begin{bmatrix}1\\0\end{bmatrix},\ t_3=0\right\},\ \left\{\mathbf{p}_4=\begin{bmatrix}1\\1\end{bmatrix},\ t_4=0\right\}.$

Реалізувати алгоритм Мамдані для цієї функції.

25. Завдання 24 для алгоритму нечіткого виведення Ларсена.

26. Завдання 24 для алгоритму нечіткого виведення Сугено.

Аудиторна робота

Приклад 3.1. Розглянемо апроксимацію логічної функції «AND» (*кон'юнкція*). Маємо таку множину пар «вхід – значення функції»: $\left\{\mathbf{p}_1=\begin{bmatrix}0\\0\end{bmatrix},\ t_1=0\right\},\ \left\{\mathbf{p}_2=\begin{bmatrix}0\\1\end{bmatrix},\ t_2=0\right\},\ \left\{\mathbf{p}_3=\begin{bmatrix}1\\0\end{bmatrix},\ t_3=0\right\},\ \left\{\mathbf{p}_4=\begin{bmatrix}1\\1\end{bmatrix},\ t_4=1\right\}.$ Реалізувати алгоритм Мамдані для цієї функції в точці $\mathbf{p}=[p_1\ p_2]^T=[0{,}1\ \ 0{,}1]^T$.

Розв'язання. Маємо базу правил вигляду:

R_1: if $p_1 = A^1_1$ and $p_2 = A^2_1$ then $y = B_1$;

R_2: if $p_1 = A^1_1$ and $p_2 = A^2_2$ then $y = B_1$;

R_3: if $p_1 = A^1_2$ and $p_2 = A^2_1$ then $y = B_1$;

R_4: if $p_1 = A^1_2$ and $p_2 = A^2_2$ then $y = B_2$.

Виконаємо фазифікацію, отримаємо такі нечіткі множини:

$p_1\in[0;\ 1]$	0	0,1	1	Рівняння прямої
$A^1_1(p_1)$	1	0,9	0	$\mu_A(p_1) = 1- p_1$
$A^1_2(p_1)$	0	0,1	1	$\mu_A(p_1) = p_1$

$p_2\in[0;\ 1]$	0	0,1	1	Рівняння прямої
$A^2_1(p_2)$	1	0,9	0	$\mu_A(p_2) = 1- p_2$
$A^2_2(p_2)$	0	0,1	1	$\mu_A(p_2) = p_2$

$y\in[0;\ 1]$	0	0,1	0,3	0,6	0,9	1	Рівняння прямої
$B_1(y)$	1	0,9	0,7	0,4	0,1	0	$\mu_{B1}(y) = 1- y$
$B_2(y)$	0	0,1	0,3	0,6	0,9	1	$\mu_{B2}(y) = y$

Рівень запуску та висновок кожного правила в точці $\mathbf{x}^*=[0{,}1\ 0{,}1]^T$ складає:

1) для першого правила маємо

$$\alpha_1 = \min\left\{A_1^1(x_1^*), A_1^2(x_2^*)\right\} = \min\left\{0{,}9; 0{,}9\right\} = 0{,}9;$$

вихід правила має вигляд нечіткої множини B^*_{11}

y	0	0,1	0,3	0,6	0,9	1
B^*_{11}	0,9	0,9	0,7	0,4	0,1	0

2) для другого правила маємо

$$\alpha_2 = \min\left\{A_1^1(x_1^*), A_2^2(x_2^*)\right\} = \min\{0,9;0,1\} = 0,1;$$

вихід правила має вигляд нечіткої множини B^*_{12}

y	0	0,1	0,3	0,6	0,9	1
B^*_{12}	0,1	0,1	0,1	0,1	0,1	0

3) для третього правила маємо

$$\alpha_3 = \min\left\{A_2^1(x_1^*), A_1^2(x_2^*)\right\} = \min\{0,1;\ 0,9\} = 0,1;$$

вихід правила має вигляд нечіткої множини B^*_{13}

y	0	0,1	0,3	0,6	0,9	1
B^*_{13}	0,1	0,1	0,1	0,1	0,1	0

4) для четвертого правила маємо

$$\alpha_4 = \min\left\{A_2^1(x_1^*), A_2^2(x_2^*)\right\} = \min\{0,1;\ 0,1\} = 0,1;$$

вихід правила має вигляд нечіткої множини B^*_2

y	0	0,1	0,3	0,6	0,9	1
B^*_2	0	0,1	0,1	0,1	0,1	0,1

Маємо $B^*_{12} = B^*_{13}$, тому вихід системи $B^* = B^*_{11} \vee B^*_{12} \vee B^*_2$ має вигляд такої нечіткої множини:

y	0	0,1	0,3	0,6	0,9	1
B^*	0,9	0,9	0,7	0,4	0,1	0,1

Маємо такий чіткий вихід моделі:

$$y^* = \frac{0\times0,9 + 0,1\times0,9 + 0,3\times0,7 + 0,6\times0,4 + 0,9\times0,1 + 1\times0,1}{0,9 + 0,7 + 0,4 + 0,1 + 0,1} = \frac{0,73}{2,2}.$$

Відповідь: вихід нечіткої моделі Мамдані $y \approx 0,332$.

Приклад 3.2. Розглянемо механізм логічного виведення Сугено. Нехай задана база правил:

R_1: if x_1 = SMALL and x_2 = BIG then $y = x_1 - x_2$

R_2: if x_1 = BIG and x_2 = SMALL then $y = x_1 + x_2$

R_3: if x_1 = BIG and x_2 = BIG then $y = x_1 + 2x_2$

Тут функції належності SMALL та BIG мають такий вигляд:

$$SMALL(v) = \begin{cases} 1, & \text{якщо } v \le 1; \\ 1 - \dfrac{v-1}{4}, & \text{якщо } 1 < v \le 5; \\ 0 & \text{інакше;} \end{cases} \quad BIG(u) = \begin{cases} 1, & \text{якщо } u \ge 5; \\ 1 - \dfrac{5-u}{4}, & \text{якщо } 1 < u \le 5; \\ 0 & \text{інакше.} \end{cases}$$

Нехай вхід має вигляд $\mathbf{x}=\begin{bmatrix}3\\3\end{bmatrix}$. Визначити вихід нечіткої моделі Сугено.

Розв'язання. Рівень запуску (роботи) та вихід першого, другого та третього правил складає:

$\alpha_1 = \min\{small(3), big(3)\} = \min\{0,5; 0,5\} = 0,5$; $y_1 = x_1 - x_2 = 3 - 3 = 0$;

$\alpha_2 = \min\{big(3), small(3)\} = \min\{0,5; 0,5\} = 0,5$; $y_2 = x_1 + x_2 = 3 + 3 = 6$;

$\alpha_3 = \min\{big(3), big(3)\} = \min\{0,5; 0,5\} = 0,5$; $y_3 = x_1 + 2x_2 = 3 + 6 = 9$.

Вихід системи має вигляд: $y = \dfrac{0 \times 0,5 + 6 \times 0,5 + 9 \times 0,5}{1,5} = 5$.

Відповідь: вихід нечіткої моделі Сугено $y = 5$.

Приклад 3.3. Розглянемо апроксимацію логічної функції «AND» (*кон'юнкція*). Маємо таку множину пар «вхід – значення функції»: $\left\{\mathbf{p}_1 = \begin{bmatrix}0\\0\end{bmatrix},\ t_1 = 0\right\}$, $\left\{\mathbf{p}_2 = \begin{bmatrix}0\\1\end{bmatrix},\ t_2 = 0\right\}$, $\left\{\mathbf{p}_3 = \begin{bmatrix}1\\0\end{bmatrix},\ t_3 = 0\right\}$, $\left\{\mathbf{p}_4 = \begin{bmatrix}1\\1\end{bmatrix},\ t_4 = 1\right\}$. Реалізу-вати алгоритм Сугено для цієї функції в точці $\mathbf{p}=[p_1\ \ p_2]^{\mathrm{T}}=[0{,}1\ \ 0{,}1]^{\mathrm{T}}$.

Розв'язання. Маємо базу правил вигляду:

R_1: if $p_1 = A^1{}_1$ and $p_2 = A^2{}_1$ then $y = 0{,}5p_1 + 0{,}5p_2$;

$\mathbf{P} = \begin{bmatrix}1 & 1\\ 0,5 & 0\\ 0,5 & 0\end{bmatrix}$; $\mathbf{P}^+ = (\mathbf{P}^{\mathrm{T}}\mathbf{P})^{-1}\mathbf{P}^{\mathrm{T}} = \begin{bmatrix}0 & 1 & 1\\ 1 & -1 & -1\end{bmatrix}$; $\mathbf{y}=[0{,}5\ 0]^{\mathrm{T}}$;

$\mathbf{y}^{\mathrm{T}}\mathbf{P}^+ = (\mathbf{P}^{\mathrm{T}}\mathbf{P})^{-1}\mathbf{P}^{\mathrm{T}} = \begin{bmatrix}0 & 0,5 & 0,5\end{bmatrix}$

R_2: if $p_1 = A^1{}_1$ and $p_2 = A^2{}_2$ then $y = 0.3333 + 0.6667p_1 - 0.3333p_2$;

$\mathbf{P} = \begin{bmatrix}1 & 1\\ 0,5 & 0\\ 0,5 & 1\end{bmatrix}$; $\mathbf{y}=[0{,}5\ 0]^{\mathrm{T}}$; $\mathbf{y}^{\mathrm{T}}\mathbf{P}^+ = (\mathbf{P}^{\mathrm{T}}\mathbf{P})^{-1}\mathbf{P}^{\mathrm{T}} = \begin{bmatrix}0,3333 & 0,6667 & -0,3333\end{bmatrix}$

R_3: if $p_1 = A^1{}_2$ and $p_2 = A^2{}_1$ then $y = 0.3333 - 0.3333p_1 + 0.6667p_2$;

$\mathbf{P} = \begin{bmatrix}1 & 1\\ 0,5 & 1\\ 0,5 & 0\end{bmatrix}$; $\mathbf{y}=[0{,}5\ 0]^{\mathrm{T}}$; $\mathbf{y}^{\mathrm{T}}\mathbf{P}^+ = (\mathbf{P}^{\mathrm{T}}\mathbf{P})^{-1}\mathbf{P}^{\mathrm{T}} = \begin{bmatrix}0,3333 & -0,3333 & 0,6667\end{bmatrix}$

R_4: if $p_1 = A^1{}_2$ and $p_2 = A^2{}_2$ then $y = 0 + 0{,}5p_1 + 0{,}5p_2$.

$\mathbf{P} = \begin{bmatrix}1 & 1\\ 0,5 & 1\\ 0,5 & 1\end{bmatrix}$; $\mathbf{P}^+ = (\mathbf{P}^{\mathrm{T}}\mathbf{P})^{-1}\mathbf{P}^{\mathrm{T}} = \begin{bmatrix}2 & -1 & -1\\ -1 & 1 & 1\end{bmatrix}$; $\mathbf{y}=[0{,}5\ 1]^{\mathrm{T}}$;

$\mathbf{y}\mathbf{P}^+ = (\mathbf{P}^{\mathrm{T}}\mathbf{P})^{-1}\mathbf{P}^{\mathrm{T}} = \begin{bmatrix}0 & 0,5 & 0,5\end{bmatrix}$.

Із прикл. 3.1 маємо:

R_1: $\alpha_1 = \min\left\{A_1^1(x_1^*), A_1^2(x_2^*)\right\} = \min\{0,9; 0,9\} = 0,9$;

$y_1 = 0{,}5p_1 + 0{,}5p_2 = 0{,}5*0{,}1 + 0{,}5*0{,}1 = 0{,}1$

R_2: $\alpha_2 = \min\left\{A_1^1(x_1^*), A_2^2(x_2^*)\right\} = \min\{0,9; 0,1\} = 0,1$;

y_2 = 0.3333+0.6667*0,1 – 0.3333*0,1=0,36664;

R_3: $\alpha_3 = \min\left\{A_2^1(x_1^*), A_1^2(x_2^*)\right\} = \min\left\{0,1;\ 0,9\right\} = 0,1$;

y_3 = 0.3333– 0.3333*0,1+0.6667*0,1=0,36664;

R_4: $\alpha_4 = \min\left\{A_2^1(x_1^*), A_2^2(x_2^*)\right\} = \min\left\{0,1;\ 0,1\right\} = 0,1$;

y_1=0,5p_1 + 0,5p_2=0,5*0,1+0,5*0,1=0,1

Вихід системи має вигляд (прикл. 3.2):

$$y = \frac{0,9\times 0,1 + 0,36664\times 0,1 + 0,36664\times 0,1 + 0,1\times 0,1}{1,2} = \frac{0,173328}{1,2} = 0,14444.$$

Відповідь: вихід нечіткої моделі $y = 0,144$.

Приклад 4.4. Нехай невідомо відображення $y = f(x_1, x_2)$, яке реалізує нечітка система, але задано множину навчання $\left\{\left(\mathbf{x} = \begin{bmatrix}1\\1\end{bmatrix},\ t = 1\right);\ \left(\mathbf{x} = \begin{bmatrix}2\\2\end{bmatrix},\ t = 2\right)\right\}$, де t – бажаний вихід системи. При моделюванні відображення $f(x_1, x_2)$ використана така база правил:

R1: if x_1 = SMALL and x_2 = SMALL then $y = ax_1 - bx_2$
R1: if x_1 = SMALL and x_2 = BIG then $y = ax_1 + bx_2$
R2: if x_1 = BIG and x_2 = SMALL then $y = bx_1 + ax_2$
R3: if x_1 = BIG and x_2 = BIG then $y = bx_1 - ax_2$

Тут a, b – невідомі параметри; а функції належності нечітких чисел SMALL та BIG мають вигляд:

$$small(v) = \begin{cases} 1 - \frac{v}{2}, & \text{якщо } 0 < v \le 2; \\ 0 & \text{інакше,} \end{cases} \quad big(v) = \begin{cases} 1 - \frac{2 - v}{2}, & \text{якщо } 0 < v \le 2; \\ 0 & \text{інакше.} \end{cases}$$

Визначити похибку апроксимації $E_1(a, b)$ та $E_2(a, b)$ для першої та другої пар навчання.

Розв'язання: Крок 1. Нехай $\mathbf{x} = \begin{bmatrix}1\\1\end{bmatrix}$ – вхід нечіткої системи. Визначимо рівні роботи правил:

$\alpha_1 = small(1) \wedge small(1) = 0,5$; $\alpha_2 = small(1) \wedge big(1) = 0,5$;

$\alpha_3 = big(1) \wedge small(1) = 0,5$; $\alpha_4 = big(1) \wedge big(1) = 0,5$.

Тому вихід системи має вигляд: $4y_1 = 2a + 2b$ або $y_1 = \frac{a+b}{2}$. Визначимо похибку навчання $E_1(a,b) = \frac{1}{2}\left(\frac{a+b}{2} - 1\right)^2$.

Крок 2. Нехай $\mathbf{x} = \begin{bmatrix}2\\2\end{bmatrix}$ – вхід нечіткої системи, маємо

$\alpha_1 = small(2) \wedge small(2) = 0$; $\alpha_2 = small(2) \wedge big(2) = 0$;

$\alpha_3 = big(2) \wedge small(2) = 0$; $\alpha_4 = big(2) \wedge big(2) = 1$.

Тому вихід системи має вигляд: $y_2 = 2b - 2a$. Визначимо похибку навчання $E_2(a,b) = \frac{1}{2}(2b - 2a - 2)^2$.

Відповідь: Похибки навчання для першої та другої пар даних $E_1(a,b) = \frac{1}{2}\left(\frac{a+b}{2} - 1\right)^2$, $E_2(a,b) = \frac{1}{2}(2b - 2a - 2)^2$.

Теоретичні відомості

1. Алгоритми нечіткого виведення [2, с. 62–74].

Компоненти нечітких моделей на основі нечеткої логіки можуть бути реалізовані по-різному. *У загальному випадку нечітке логічне виведення реалізують у вигляді таких чотирьох кроків*:

Крок 1. Будують ФН, які визначають ступінь (міру) істинності передумови кожного правила.

Крок 2. Виконують логічне виведення, яке полягає в тому, що виходячи із значень істинності передумов правила, обчислюють висновок кожного правила.

Крок 3. Виконують композицію всіх нечітких підмножин, які відповідають кожній змінній виведення.

Крок 4. Виконують дефазифікацію множини нечітких висновків в чітке число.

Сукупність окремих реалізацій компонентів нечіткої моделі визначає *алгоритм нечіткого виведення.* Розглянемо найпоширеніші в наш час алгоритми нечіткого виведення (Мамдані – Mamdani, Ларсена – Larsen, Такагі–Сугено – Takagi–Sugeno, алгоритм на основі нечіткої моделі з адаптацією операцій над нечіткими множинами).

1. Алгоритм нечіткого виведення Мамдани (Mamdani, рис. 3.1). Концепція лінгвістичної нечіткої моделі, яка відтворює людський спосіб мислення, була запропонована в перших роботах Заде. Ідея застосування даної концепції до нечіткого управління динамічними об'єктами належить Мамдані, який запропонував спосіб побудови моделі дій людини-оператора, керуючого об'єктом. В наш час цей метод використовується найчастіше, хоча були розроблені й інші типи моделей.

У рамках методу Мамдані система, яка моделюється, розглядається як чорний ящик, що характеризується недостатністю інформації про фізичні явища, що відбуваються в ній. Метою є розробка моделі, яка виконує таке відображення своїх входів (вектор **x**) у вихід Y (обмежимося розглядом систем з одним виходом), яке забезпечувало б якомога точнішу апроксимацію реальної системи (наприклад, в сенсі середньої абсолютної похибки). Зазначене відображення припускає існування деякої геометричної поверхні, яку будемо далі називати поверхнею відображення, в просторі, що задається декартовим добутком X×Y.

Модель Мамдані включає в себе множину правил, де кожне правило задає у визначеному просторі деяку нечітку точку. На основі множини нечітких точок формується нечіткий графік, в якому механізм інтерполяції між точками залежить від використаного апарату нечіткої логіки.

Нехай Базу знань складають два нечітких правила:

R_1: if x=A_1 and y=B_1 then z=C_1;
R_2: if x=A_2 and y=B_2 then z=C_2.

Нечітке логічне виведення на основі алгоритму Мамдані охоплює такі кроки:

Крок 1 (фазифікація). Знаходимо міри істинності передумов кожного правила для заданих значень вхідних змінних $\mu_{A1}(x_0)$, $\mu_{A2}(x_0)$, $\mu_{B1}(y_0)$, $\mu_{B2}(y_0)$.

Крок 2 (агрегування: Т-норма – min-кон'юнкція). Декартовий добуток нечітких множин задано виразом: $\mu_{A_1\times B_1}(x,y)=\min\left(\mu_{A_1}(x),\mu_{B_1}(y)\right)$.

Виконуємо агрегування ступенів істинности передумов кожного правила (або знаходимо рівні «відсікання» α_1, α_2 для передумов кожного з правил):

$$\alpha_1 = A_1(x_0)\wedge B_1(y_0)=\min\left(\mu_{A_1}(x_0),\mu_{B_1}(y_0)\right);$$

$$\alpha_2 = A_2(x_0)\wedge B_2(y_0)=\min\left(\mu_{A_2}(x_0),\mu_{B_2}(y_0)\right).$$

Крок 3 (нечітка імплікація – операція min-кон'юнкції).

Виконуємо активацію висновків правил: процедура активації висновків нечітких правил полягає у визначенні модифікованих ФН цих висновків для кожного правила $\mu_{Ci}(z)$ на основі виконання *операції композиції*, модифікованої для нечіткого правила, між визначеним на попередньому етапі значенням агрегованих ступенів істинності передумов цього правила α_i і відповідною ФН його висновку $\mu_{Ci}(z)$. Таким чином, значення ФН висновків мають вигляд: $\mu_{C_1'}(z) = (\alpha_1 \wedge C_1(z)) = \min(\alpha_1, \mu_{C_1}(z))$;

$\mu_{C_2'}(z) = (\alpha_2 \wedge C_2(z)) = \min(\alpha_2, \mu_{C_2}(z))$. Отримані таким чином результати корегують шляхом їх алгебраїчного добутку на вагові коефіцієнти відповідних правил. Якщо ці коефіцієнти не задані, то передбачається, що вони дорівнюють одиниці.

Крок 4 (акумулювання активованих висновків правил – max-диз'юнкція). Після отримання активованих висновків для кожної вихідної змінної кожного з нечітких правил виконується процедура їх акумулювання. Результат такого акумулювання для вихідних змінних знаходять шляхом об'єднання отриманих на попередньому етапі відповідних нечітких множин.

Виконуємо об'єднання знайдених функцій і визначаємо результуючу нечітку множину для ФН вихідної змінної таким чином:

$$\mu_{\sum}(z) = C(z) = C'_1(z) \vee C'_2(z) = (\alpha_1 \wedge C_1(z)) \vee (\alpha_2 \wedge C_2(z)) = \max(\mu_{C_1'}(z), \mu_{C_2'}(z)).$$

Крок 5 (дефазифікація). Приведення до чіткості полягає у перетворенні нечітких значень знайдених вихідних змінних в чіткі. Виконуємо дефазифікацію, наприклад, методом центру ваги та знаходимо чітке значення:

$z_0 = \dfrac{\int_{\underline{z}}^{\overline{z}} z\mu_0(z)dz}{\int_{\underline{z}}^{\overline{z}} \mu_0(z)dz}$, де інтервал $[\underline{z}, \overline{z}]$ є носієм ФН вихідної змінної.

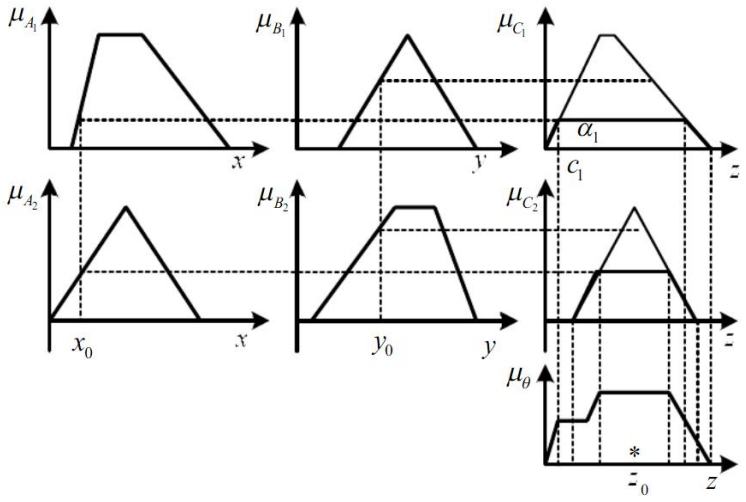


Рис. 3.1. Реалізація алгоритму Мамдані

Алгоритм Ларсена (Larsen, рис. 3.2)

В алгоритмі нечіткого виведення Ларсена маємо:

1) база правил формується аналогічно алгоритму Мамдані;

2) t-норма – зазвичай min-кон'юнкція; декартовий добуток нечітких множин задано виразом: $\mu_{A_1\times B_1}(x,y)=\min\left(\mu_{A_1}(x),\mu_{B_1}(y)\right)$;

3) нечітка імплікація – нечітке множення;

4) акумулювання активованих висновків правил – max-диз'юнкція.

Нечітке логічне виведення на основі алгоритму Ларсена охоплює такі кроки:

Крок 1 (фазифікація). Визначаємо значення ступенів істинності передумов кожного з правил $A_1(x_0)$, $A_2(x_0)$, $B_1(y_0)$, $B_2(y_0)$.

Крок 2 (активація висновків кожного із нечітких правил).

1) обчислюємо значення рівнів відсікання α_i (min-кон'юнкція або алгебраїчний, граничний, драстичний добутки)**:**

- алгебраїчний добуток ступенів істинності передумов**:**

$$\alpha_1=A_1(x_0)\wedge B_1(y_0)=\mathrm{prod}\left(\mu_{A_1}(x_0),\mu_{B_1}(y_0)\right);$$

$$\alpha_2=A_2(x_0)\wedge B_2(y_0)=\mathrm{prod}\left(\mu_{A_2}(x_0),\mu_{B_2}(y_0)\right).$$

- min- кон'юнкція:

$$\alpha_1 = \min\left(\mu_{A_1}(x_0), \mu_{B_1}(y_0)\right); \quad \alpha_2 = \min\left(\mu_{A_2}(x_0), \mu_{B_2}(y_0)\right)$$

- граничний добуток: $f_{A_i \wedge B_i} = \max\left(\mu_{A_i}(x_0) + \mu_{B_i}(y_0) - 1, 0\right);$

$$\alpha_1 = f\left(\mu_{A_1}(x_0), \mu_{B_1}(y_0)\right); \quad \alpha_2 = f\left(\mu_{A_2}(x_0), \mu_{B_2}(y_0)\right)$$

- драстичний добуток: $f_{A_i \wedge B_i} = \begin{cases} \mu_{A_i}(x_0), \text{якщо } \mu_{A_i}(x_0) = 1; \\ \mu_{B_i}(y_0), \text{якщо } \mu_{B_i}(y_0) = 1; \\ 0, \text{в інших випадках.} \end{cases}$

$$\alpha_1 = f\left(\mu_{A_1}(x_0), \mu_{B_1}(y_0)\right); \quad \alpha_2 = f\left(\mu_{A_2}(x_0), \mu_{B_2}(y_0)\right)$$

2) виконуємо активацію висновків кожного правила $\alpha_1 C_1(z)$, $\alpha_2 C_2(z)$:

$$\mu_{C_1'}(z) = \left(\alpha_1 \wedge C_1(z)\right) = \text{prod}\left(\alpha_1, \mu_{C_1}(z)\right);$$

$$\mu_{C_2'}(z) = \left(\alpha_2 \wedge C_2(z)\right) = \text{prod}\left(\alpha_2, \mu_{C_2}(z)\right);$$

Крок 3. Знаходимо результуючу нечітку підмножину із ФН висновків: $\mu_{\sum}(z) = C'_1(z) \vee C'_2(z) = \max\left(\mu_{C_1'}(z), \mu_{C_2'}(z)\right).$

У загальному випадку маємо $\mu_{\sum}(z) = \bigcup_{i=1}^{n} \left(\alpha_i C_i(z)\right).$

Крок 4 (дефазифікація). Приведення до чіткості (як в алгоритмі Мамдані).

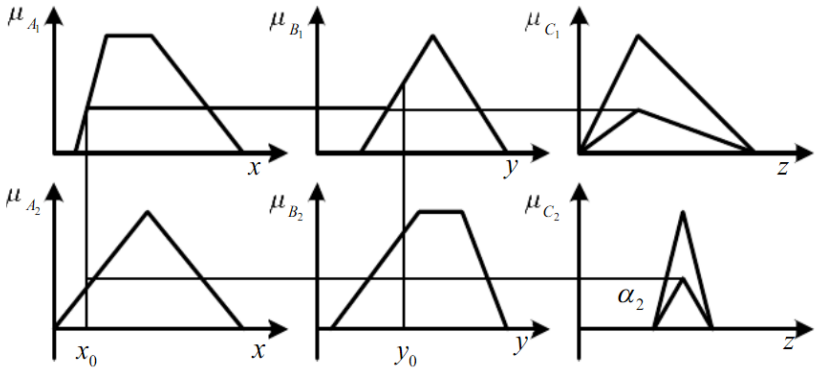


Рис. 3.2. Реалізація алгоритму Ларсена

Алгоритм Такагі – Сугено (Sugeno і Takagi, рис. 3.3**)**

Нехай використовується множина правил вигляду:

R1: if $x \in A_1$ and $y \in B_1$ then $z_1 = a_1 x + b_1 y$
R2: if $x \in A_2$ and $y \in B_2$ then $z_2 = a_2 x + b_2 y$

Нечітке логічне виведення на основі алгоритму Такагі–Сугено охоплює такі кроки:

Крок 1 (фазифікація: як в алгоритмі Мамдані). Визначаємо значення ступенів істинності передумов кожного з правил $A_1(x_0)$, $A_2(x_0)$, $B_1(y_0)$, $B_2(y_0)$,

Крок 2. Агрегування ступенів істинності передумов у кожному з правил (min-кон'юнкція або алгебраїчний, граничний, драстичний добутки): визначаємо α_1, α_2: $\alpha_1 = A_1(x_0) \wedge B_1(y_0)$; $\alpha_2 = A_2(x_0) \wedge B_2(y_0)$.

Крок 3 (активація висновків по кожному з правил): визначаємо чіткі значення вихідних змінних: $z_1^* = a_1x_0 + b_1y_0$; $z_2^* = a_2x_0 + b_2y_0$.

Етап акумулювання активованих висновків правил в алгоритмі відсутній внаслідок чітких значень вихідних змінних.

Крок 4. Визначаємо чітке значення $z_0 = \dfrac{\alpha_1 z^*_1 + \alpha_2 z^*_2}{\alpha_1 + \alpha_2}$.

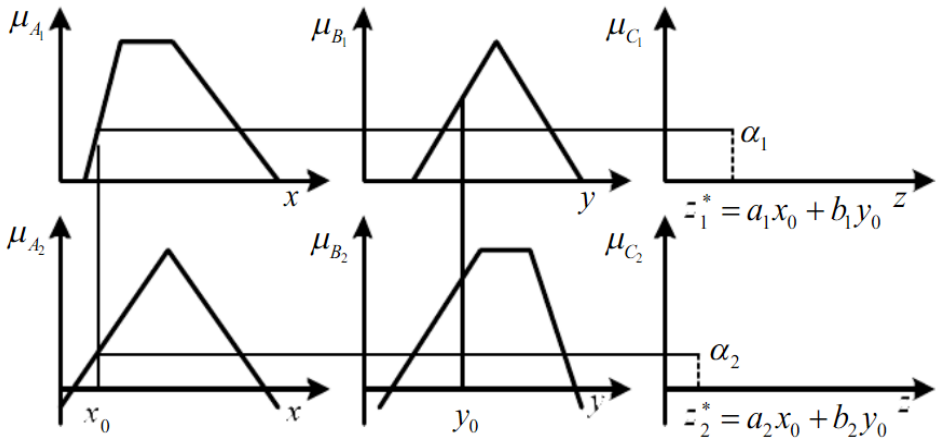


Рис. 3.3. Реалізація алгоритму Такагі–Сугено

Спрощений алгоритм (рис. 3.4)

Нехай Базу знань складають два нечітких правила

R1: if $x \in A_1$ and $y \in B_1$ then $z1=C_1$; R2: if $x \in A_2$ and $y \in B_2$ then $z2=C_2$

де C_i, i=1, 2 – чіткі числа

Крок 1. Визначаємо значення ступенів істинності передумов кожного з правил $A_1(x_0)$, $A_2(x_0)$, $B_1(y_0)$, $B_2(y_0)$.

Крок 2. Розраховуємо значення α_1, α_2 (як в алгоритмі Мамдані): $\alpha_1 = A_1(x_0) \wedge B_1(y_0)$; $\alpha_2 = A_2(x_0) \wedge B_2(y_0)$.

Як операція агрегування можуть використовуватися й інші нечіткі логічні операції t-норми.

Етап акумулювання висновків нечітких правил в алгоритмі відсутній внаслідок чітких значень вихідних змінних.

Крок 3 (дефазифікація). Визначаємо чітке число $z_0 = \dfrac{\alpha_1 C_1 + \alpha_2 C_2}{\alpha_1 + \alpha_2}$.

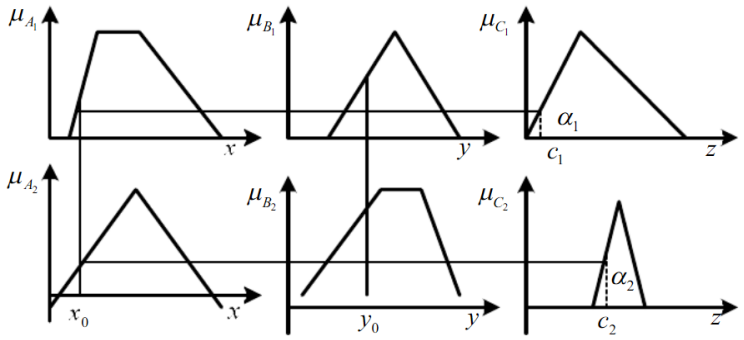


Рис. 3.4. Реалізація спрощеного алгоритму

2. Процес розробки систем нечіткого виведення (СНВ) в інтерактивному режимі [3, с. 56–65].

Для реалізації процесу нечіткого моделювання в середовищі MatLab використовується пакет Fuzzy Logic Toolbox. З його допомогою користувач може виконувати необхідні дії, що стосуються розробки та застосування нечітких моделей, працюючи в одному з наступних режимів: 1) *в інтерактивному режимі* (за допомогою графічних засобів редагування та візуалізації всіх компонентів системи нечіткого виведення); 2) *в режимі команд* (за допомогою введення імен відповідних функцій та їхніх аргументів безпосередньо у вікно команд системи MatLab). Для розробки та подальшого використання СНВ в інтерактивному режимі використовують такі графічні засоби, які входять до складу пакету Fuzzy Logic Toolbox:

1) редактор систем нечіткого виведення FIS (FIS Editor);

2) редактор функцій належності СНВ (Membership Function Editor);

3) редактор правил СНВ (Rule Editor);

4) програма перегляду правил СНВ (Rule Viewer);

5) програма перегляду поверхні СНВ (Surface Viewer).

Наведемо *алгоритм розв'язання задачі апроксимації функції за допомогою системи нечіткого виведення в інтерактивному режимі*.

Перший крок: 1) відкрити редактор СНВ FIS; 2) командою Add input меню Edit додати до системи вхідну змінну (якщо потрібно); 3) ввести позначення вхідних та вихідних змінних. Після виконання першого етапу отримаємо структуру системи нечіткого виведення.

Другий крок (процес фаззифікації вхідних та вихідних змінних):

– відкрити редактор функцій належності;

– командою Add MFs меню Edit задати необхідну кількість термів та визначити функції належності (наприклад, *gaussmf*);

– ввести діапазон можливих значень для вхідних та вихідних змінних.

Третій крок: 1) відкрити редактор бази знань; 2) проаналізувавши отримані дані, згенерувати базу знань.

Четвертий крок: дослідити механізм нечіткого логічного виведення за допомогою команди Rules меню View.

2.1. Редактор систем нечіткого виведення FIS. Редактор FIS можна активувувати за допомогою введення функції *fuzzy* або *fuzzy('lab2')* у вікні команд. Цей редактор надає користувачеві можливість задавати та редагувати такі властивості СНВ, як число вхідних і вихідних змінних, тип СНВ, метод дефаззификації тощо.

Якщо виконується функція *fuzzy*, то редактор FIS викликається для системи нечіткого виведення з ім'ям *Untitled* (рис. 3.5). При цьому за замовчуванням задаються такі параметри, як тип системи нечіткого виведення (Мамдані), нечіткі логічні операції, методи імплікації, агрегування та дефаззифікації тощо. Користувач може погодитися з цими значеннями або змінити їх. Якщо функцію *fuzzy* викликають в формі *fuzzy* ('*lab2*'), де *lab*2 – ім'я зовнішнього файлу з росширенням fis, що містить опис розробленої раніше

СНВ, то редактор FIS активується разом із завантаженою системою з ім'ям *lab*2.

Редактор FIS має власний графічний інтерфейс і дозволяє викликати всі інші редактори та програми перегляду СНВ. У верхній частині його робочого інтерфейсу міститься діаграма, яка зображує входи та виходи СНВ, в центрі яких знаходиться так званий *процесор нечітких правил.* Активація одного з прямокутників, які зображують вхідний або вихідний аргументи, робить відповідну змінну поточною. Прямокутник поточної змінної при цьому виділяється червоним кольором. За допомогою зазначених прямокутників можна викликати редактор ФН, а за допомогою процесора нечітких правил – редактор правил для відповідної СНВ. Редактор FIS має також головне меню, призначення опцій якого наведено в Додатку А.

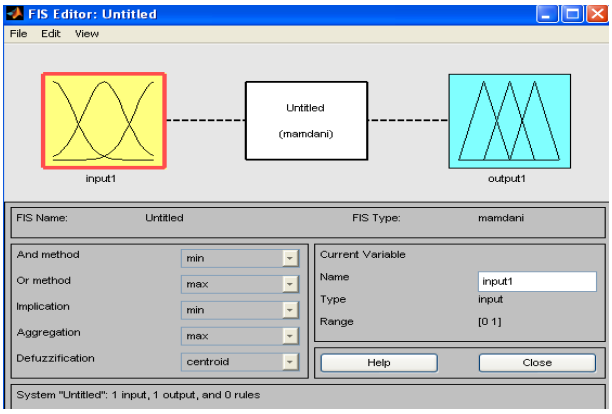


Рис. 3.5. Графічний інтерфейс редактора FIS, який викликають командою *fuzzy*

2.2. Редактор функцій належності призначений для визначення функцій належності окремих термів СНВ в графічному режимі. Його можна активувати за допомогою введення функції *mfedit*, або *mfedit* (*'a'*) (*mfedit* (*a*)) у вікні команд, а також, використовуючи головне меню редактора FIS (команда Membership Functions меню Edit).

Функція mfedit викликає редактор ФН без завантаження СНВ. При застосуванні функції *mfedit ('a')* викликається редактор функцій належності, який дозволяє користувачеві в графічному режимі аналізувати та модифікувати всі функції належності деякої структури FIS, яка зберігається в зовнішньому файлі з ім'ям *a.fis*. При цьому ім'я, тип і параметри кожної функції належності

можна змінювати. Функція *mfedit (a)*, працює зі змінною робочого простору MatLab, яка відповідає структурі FIS з ім'ям *a*.

Редактор ФН надає користувачеві не тільки можливість обрати будь-яку з вбудованих функцій належності, але й задати свою власну. Результат виклику редактора функцій належності за допомогою функції *mfedit* ('*lab*2') зображено на рис. 3.6. Для відображення графіків ФН потрібно обрати необхідну змінну в лівій частині графічного інтерфейсу редактора під заголовком FIS Variables. Щоб обрати потрібну ФН, треба натиснути на ній в основному вікні з графіками ФН. Редактор ФН має головне меню, яке дозволяє користувачеві викликати інші графічні засоби роботи з СНВ, завантажувати та зберігати структуру FIS у зовнішніх файлах тощо.

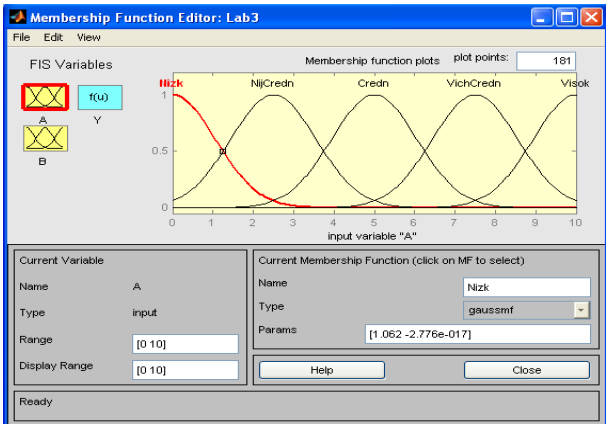


Рис. 3.6. Редактор ФН, який викликається функцією *mfedit*('*lab*2')

2.3. Редактор правил СНВ. Редактор правил СНВ призначений для визначення окремих правил СНВ в графічному режимі. Його можна активувати за допомогою введення функції *ruleedit*(*'a'*) (або *ruleedit* (*a*)) у вікні команд, а також, використовуючи головне меню редактора FIS (команда Rules меню Edit). Функція *ruleedit* (*'a'*) викликає редактор правил, який дозволяє користувачеві в графічному режимі аналізувати та модифікувати правила СНВ FIS, збереженої в зовнішньому файлі з ім'ям *a.fis*. Ця функція дозволяє також виконувати граматичний аналіз правил, які використовуються в даній системі. Функція *ruleedit*(*a*) викликає редактор правил для змінної робочого простору MatLab, яка відповідає структурі FIS з ім'ям *a*.

Щоб використовувати редактор правил для створення відповідних правил, необхідно попередньо визначити всі вхідні та вихідні змінні. Для цього можна скористатися редактором FIS і редактором ФН. При цьому задавати правила можна за допомогою вибору відповідних значень термів вхідних і вихідних змінних. Результат виконання функції *ruleedit* (‘*lab*2’) зображено на рис. 3.7. Редактор правил СНВ має головне меню, призначення пунктів якого описане в Додатку А.

Для текстового запису правил нечітких продукцій використовуються службові слова "if", "then", "is", "AND", "OR" тощо. При записі їх у символьній формі вони замінюються символами відповідних операцій.

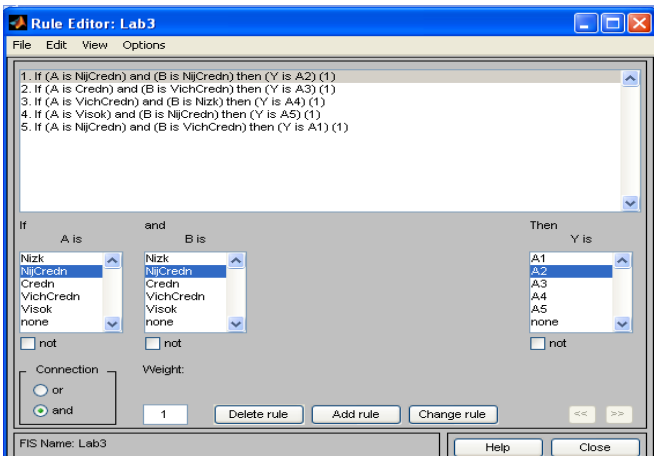


Рис. 3.7. Редактор правил, який викликається функцією *ruleedit* (‘*lab2*’)

Наприклад, правило «if (A is $\bar{A}$) and (B is $\bar{B}$) then (C is $\bar{C}$)» має вигляд: "(A = $\bar{A}$) & (B = $\bar{B}$)$\rightarrow$(C= $\bar{C}$)". Поля введення, що розміщені в середній частині графічного інтерфейсу редактора правил, дозволяють задати нове правило в СНВ. Для цього необхідно виділити ім’я терма відповідної змінної, яка попередньо має бути визначена за допомогою редактора функцій належності. Якщо терм не входить у правило, то для відповідного поля введення варто обрати значення «none». Якщо в умові правила використовують логічне заперечення деякого терма, то для цього терма слід зробити помітку «not».

Редактор правил дозволяє задати логічні зв’язки для підумов правила (перемикач Connection), а також, вагу правила (поле введення Weight). Кнопки,

які розміщені в нижній частині графічного інтерфейсу редактора правил, відповідають за видалення виділеного правила (Delete rule), додавання нового правила в систему (Add rule), внесення змін у виділене в цьому вікні правило (Change rule), виклик вбудованої довідкової системи MatLab (Help) і закриття вікна редактора правил (Close).

2.4. Програма перегляду правил СНВ. Головне призначення програми перегляду правил СНВ полягає у візуалізації результатів нечіткого виведення та отриманні значень вихідних змінних в залежності від початкових значень вхідних змінних. Її графічний інтерфейс можна завантажити за допомогою введення функції *ruleview*(*'a'*) (*ruleview*(*a*)) у вікні команд, а також, використовуючи головне меню одного з вищеописаних редакторів (команда Rules меню View). Функція *ruleview*(*'a'*) викликає програму перегляду правил, що зображує діаграму нечіткого виведення для структури FIS, збереженої в зовнішньому файлі з ім'ям *a.fis*. В свою чергу, функція *ruleview*(*a*) викликає програму перегляду правил для змінної робочого простору MatLab, яка відповідає структурі FIS з ім'ям *a*.

Програма перегляду правил не дозволяє редагувати правила та ФН термів змінних. Її зазвичай використовують після завершення розробки СНВ на етапі її аналізу та оцінки. Цю програму також доцільно використовувати в тому випадку, коли необхідно візуально зобразити весь процес нечіткого виведення. При цьому, користувач має можливість оцінити значення вихідних змінних нечіткої моделі та вплив кожного з правил на результат нечіткого виведення, змінюючи значення вхідних змінних. Програма перегляду правил має головне меню, призначення пунктів якого описується у Додатку А.

Результат виконання функції *ruleview* ('*lab*2') зображено на рис. 3.8. В центральній частині графічного інтерфейсу програми перегляду правил розташовані прямокутники, які відповідають окремим вхідним змінним (їхні ФН зображені жовтим кольором) і вихідним змінним (їхні ФН синього кольору)

правил нечіткого виведення. При цьому кожному правилу відповідає окремий рядок цих прямокутників.

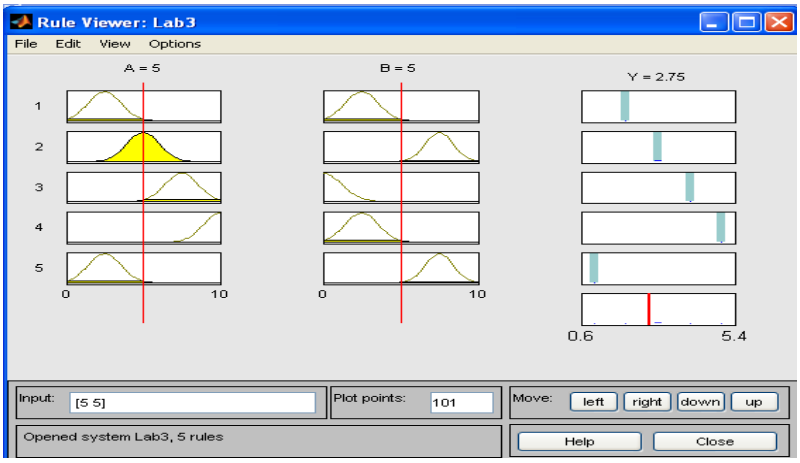


Рис. 3.8. Графічний інтерфейс програми перегляду правил, який викликається функцією *ruleview*(‘*lab*2’)

Номера правил вказані в лівій частині графічного інтерфейсу. У правій нижній частині графічнного інтерфейсу програми перегляду правил розташований прямокутник, який зображує дефаззифікацію вихідної змінної після акумулювання всіх висновків правил нечіткого виведення. Одержане в результаті дефаззифікації значення вихідної змінної вказується у верхній частині стовпчика з ім’ям цієї змінної (стовпчик з ім’ям Y=2.75 на рис. 3.9). Прямокутники, які відповідають вхідним змінним, перетинає вертикальна пряма червоного кольору, положення якої відповідає конкретному значенню вхідної змінної відповідного стовпчика. Задати конкретні значення вхідних змінних можна записавши їх у полі введення Input, або переміщуючи вертикальні прямі в потрібному напрямку за допомогою миші. Після кожної зміни значення окремої вхідної змінної система MatLab виконує процедуру нечіткого виведення та відображає відповідні значення вихідних змінних.

2.5. Програма перегляду поверхні нечіткого виведення. Її призначення полягає у наочному зображенні графіків залежності вихідних змінних від окремих вхідних змінних. Графічний інтерфейс можна відкрити за допомогою введення функції *surfview*(‘*a*’) або *surfview*(*a*) у вікні команд, а також,

використовуючи головне меню одного з редакторів: редактора FIS, редактора ФН або редактора правил (команда Surface меню View).

Функція *surfview*(‘*a*’) викликає програму перегляду поверхні нечіткого виведення для структури FIS, збереженної в файлі з ім’ям *a.fis*, для однієї або двох її вхідних змінних. Програма перегляду поверхні виведення має головне меню, призначення пунктів якого описано в Додатку А. Використовуючи головне меню цієї програми, користувач може обрати вхідні змінні, яким відповідають горизонтальні осі системи координат, а також вихідну змінну, якій відповідає вертикальна вісь координат. Результат виконання функції *surfview* (‘*lab*2’) зображено на рис. 3.9.

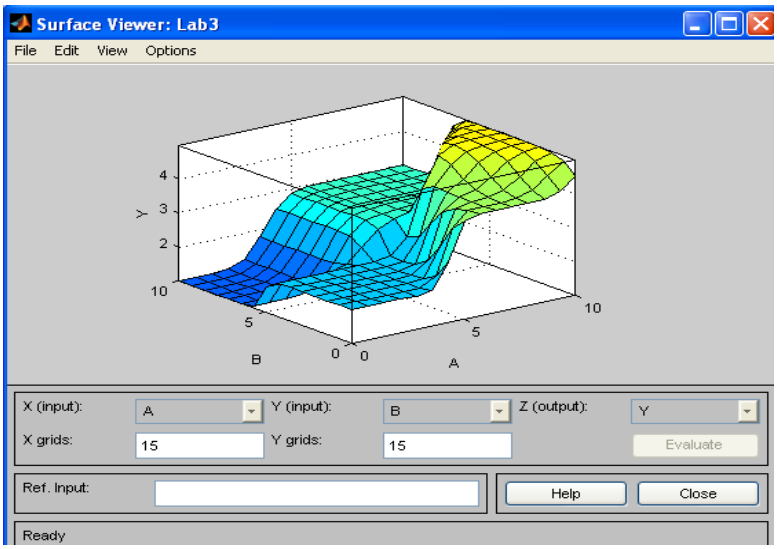


Рис. 3.9. Програма перегляду поверхні виведення, яку викликає функція *surfview* (‘*lab*2’)

3. Структура FIS. Структура FIS є об'єктом MATLAB, який містить всю інформацію про конкретну СНВ (включаючи імена змінних, визначення ФН тощо). Функції доступу getfis і setfis дозволяють вивчити цю структуру. Вона має вигляд, зображений на рис. 3.10.

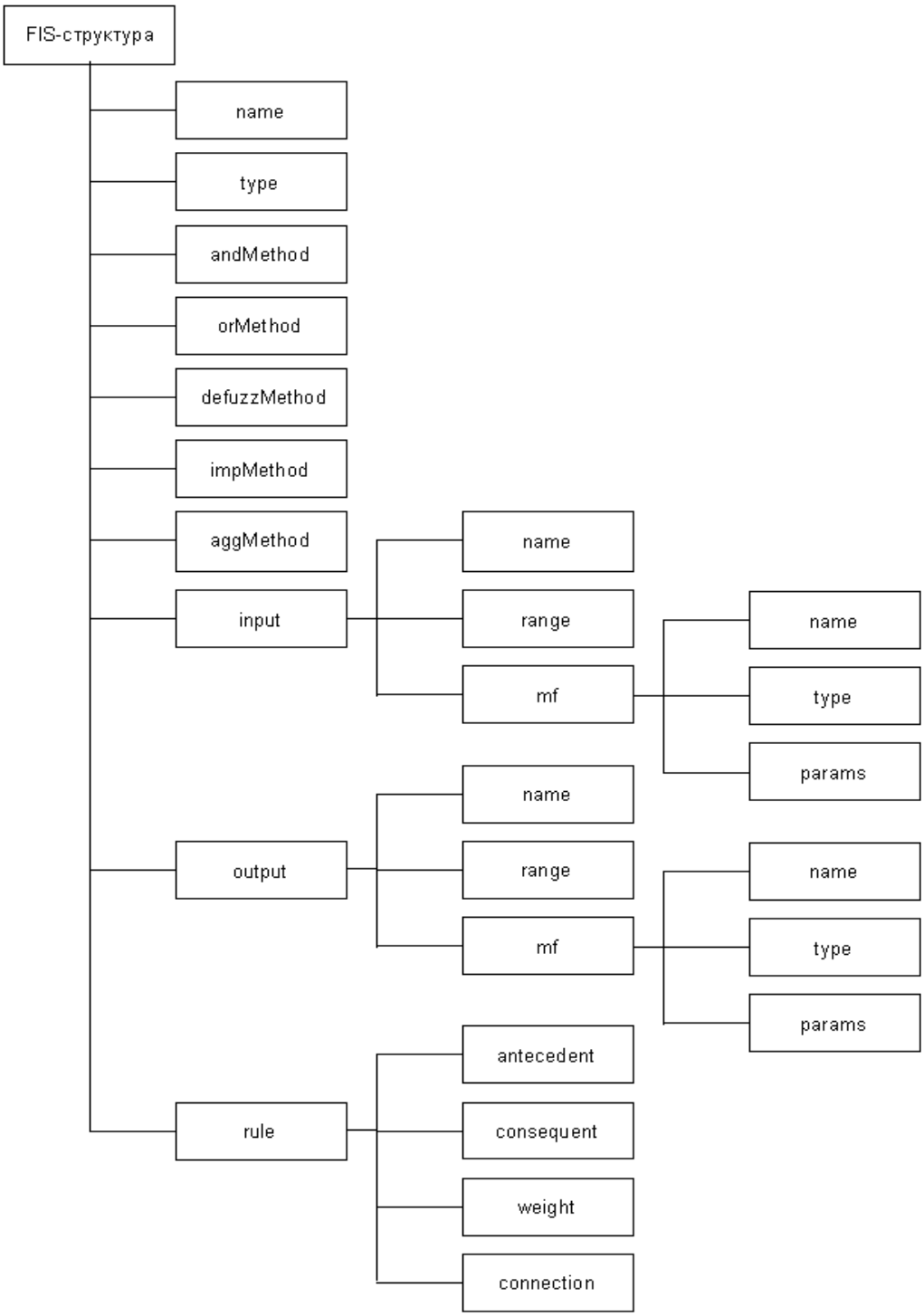


Рис. 3.10. Структура даних, яка використовується для опису СНВ в MatLab

Таблиця 3.1.

Поля структури даних системи нечіткого виведення

name	ім’я системи нечіткого виведення
type	тип системи нечіткого виведення (алгоритм)
andMethod	реалізація логічної операції “І”
orMethod	реалізація логічної операції “АБО”
defuzzMethod	метод дефазифікації
impMethod	реалізація операції імплікації
aggMethod	реалізація операції об’єднання ФН вихідної змінної
input	масив вхідних змінних
input.name	ім’я вхідної змінної
input.range	діапазон значень вхідної змінної
input.mf	масив функції належності вхідної змінної
input.mf.name	ім’я функції належності вхідної змінної
input.mf.type	модель функції належності вхідної змінної
input.mf.params	масив параметрів функції належності вхідної змінної
output	масив вихідних змінних
output.name	ім’я вихідної змінної
output.range	діапазон значень вихідної змінної
output.mf	масив функції належності вихідної змінної
output.mf.name	ім’я функції належності вихідної змінної
output.mf.type	модель функції належності вихідної змінної
output.mf.params	масив параметрів функції належності вихідної змінної
rule	масив правил нечіткої бази знань
rule.antecedent	умова правила нечіткої продукції
rule.consequent	наслідок правила нечіткої продукції
rule.weight	вага правила нечіткої продукції
rule.connection	логічний зв’язок змінних всередині правила

Заповнення полів FIS структури

```
showfis(a)
1.  Name             tipper
2.  Type             mamdani
3.  Inputs/Outputs   [ 2 1 ]
4.  NumInputMFs      [ 3 2 ]
5.  NumOutputMFs     3
6.  NumRules         3
7.  AndMethod        min
8.  OrMethod         max
9.  ImpMethod        min
10. AggMethod        max
11. DefuzzMethod     centroid
12. InLabels         service
13.                  food
14. OutLabels        tip
15. InRange          [ 0 10 ]
16.                  [ 0 10 ]
17. OutRange         [ 0 30 ]

18. InMFLabels       poor
19.                  good
20.                  excellent
21.                  rancid
22.                  delicious
23. OutMFLabels      cheap
24.                  average
25.                  generous
26. InMFTypes        gaussmf
27.                  gaussmf
28.                  gaussmf
29.                  trapmf
30.                  trapmf
31. OutMFTypes       trimf
32.                  trimf
33.                  trimf
34. InMFParams       [ 1.5 0 0 0 ]
35.                  [ 1.5 5 0 0 ]
36.                  [ 1.5 10 0 0 ]
37.                  [ 0 0 1 3 ]
38.                  [ 7 9 10 10 ]
39. OutMFParams      [ 0 5 10 0 ]
40.                  [ 10 15 20 0 ]
```

```
41.                  [ 20 25 30 0 ]
42. Rule Antecedent  [ 1 1 ]
43.                  [ 2 0 ]
44.                  [ 3 2 ]
42. Rule Consequent  1
43.                  2
44.                  3
42. Rule Weigth      1
43.                  1
44.                  1
42. Rule Connection  2
43.                  1
44.                  2
```

Список команд, пов’язаних із структурою FIS включає в себе такі: getfis, setfis, showfis, addvar, addmf, addrule, rmvar, rmmf.

Команди Fuzzy Logic Toolbox. Заповнюємо структуру нечіткої моделі такими елементами:

```
a=newfis('tipper');
a.input(1).name='service';
a.input(1).range=[0 10];
a.input(1).mf(1).name='poor';
a.input(1).mf(1).type='gaussmf';
a.input(1).mf(1).params=[1.5 0];
a.input(1).mf(2).name='good';
a.input(1).mf(2).type='gaussmf';
a.input(1).mf(2).params=[1.5 5];
a.input(1).mf(3).name='excellent';
a.input(1).mf(3).type='gaussmf';
a.input(1).mf(3).params=[1.5 10];
a.input(2).name='food';
a.input(2).range=[0 10];
a.input(2).mf(1).name='rancid';
a.input(2).mf(1).type='trapmf';
a.input(2).mf(1).params=[-2 0 1 3];
a.input(2).mf(2).name='delicious';
a.input(2).mf(2).type='trapmf';
a.input(2).mf(2).params=[7 9 10 12]
a.output(1).name='tip';
a.output(1).range=[0 30];
a.output(1).mf(1).name='cheap';
a.output(1).mf(1).type='trimf';
a.output(1).mf(1).params=[0 5 10];
a.output(1).mf(2).name='average';
a.output(1).mf(2).type='trimf';
a.output(1).mf(2).params=[10 15 20];
a.output(1).mf(3).name='generous';
```

```
a.output(1).mf(3).type='trimf';
a.output(1).mf(3).params=[20 25 30];
a.rule(1).antecedent=[1 1];
a.rule(1).consequent=[1];
a.rule(1).weight=1;
a.rule(1).connection=2;
a.rule(2).antecedent=[2 0];
a.rule(2).consequent=[2];
a.rule(2).weight=1;
a.rule(2).connection=1;
a.rule(3).antecedent=[3 2];
a.rule(3).consequent=[3];
a.rule(3).weight=1;
a.rule(3).connection=2
```

Це еквівалентно такому набору команд Fuzzy Logic Toolbox:

```
a=newfis('tipper');
a=addmf(a,'input',1,'service',[0 10]);
a=addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
a=addmf(a,'input',1,'good','gaussmf',[1.5 5]);
a=addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);
a=addvar(a,'input','food',[0 10]);
a=addmf(a,'input',2,'rancid','trapmf',[-2 0 1 3]);
a=addmf(a,'input',2,'delicious','trapmf',[7 9 10 12]);
a=addvar(a,'output','tip',[0 30]);
a=addmf(a,'output',1,'cheap','trimf',[0 5 10]);
a=addmf(a,'output',1,'average','trimf',[10 15 20]);
a=addmf(a,'output',1,'generous','trimf',[20 25 30]);
ruleList=[ ...
1 1 1 1 2
2 0 2 1 1
3 2 3 1 2];
a=addrule(a,ruleList);
```

Оцінка виходу нечіткої системи

Щоб оцінити вихід нечіткої системи для заданого входу, використовуйте функцію evalfis. Наприклад, такий набір команд оцінює вихід моделі для входу [1 2].

```
a = readfis(‘tipper');evalfis([1 2], a)
ans = 5.5586
```

Результат моделювання

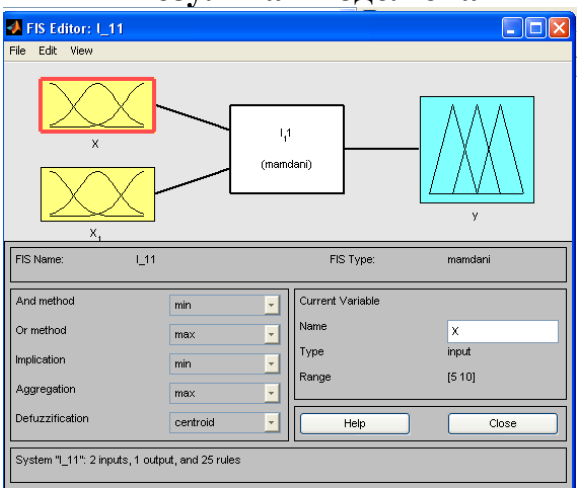
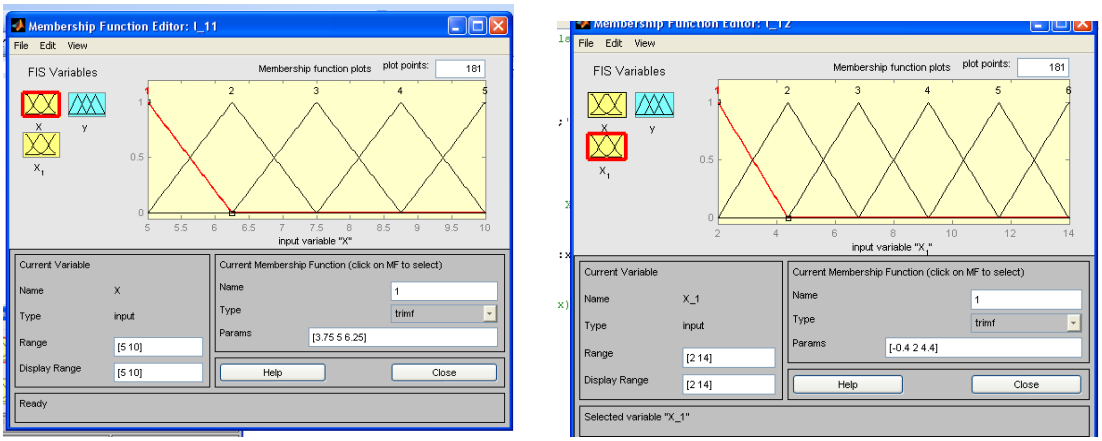


Рис. 3.11. Редактор систем нечіткого виведення FIS



а *б*

Рис. 3.12. Редактор функцій належності: функції належності вхідної змінної x_1 (*а*) та x_2 (*б*) (на рис. описані терми змінної, а також її інтервал зміни)

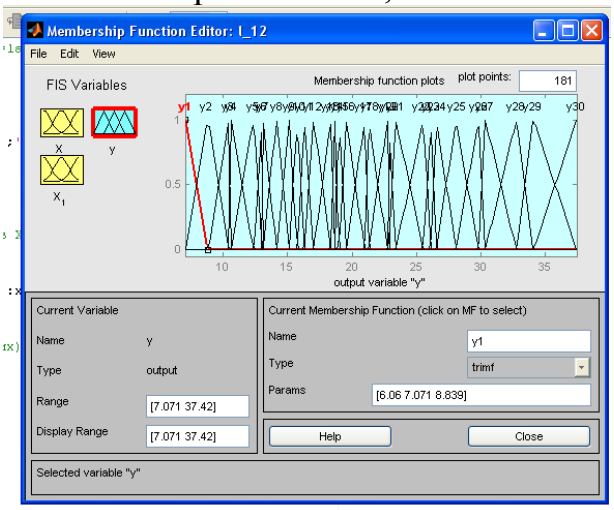


Рис.3.13. Редактор ФН: функції належності вихідної змінної *y* (терми змінної, а також її інтервал зміни)

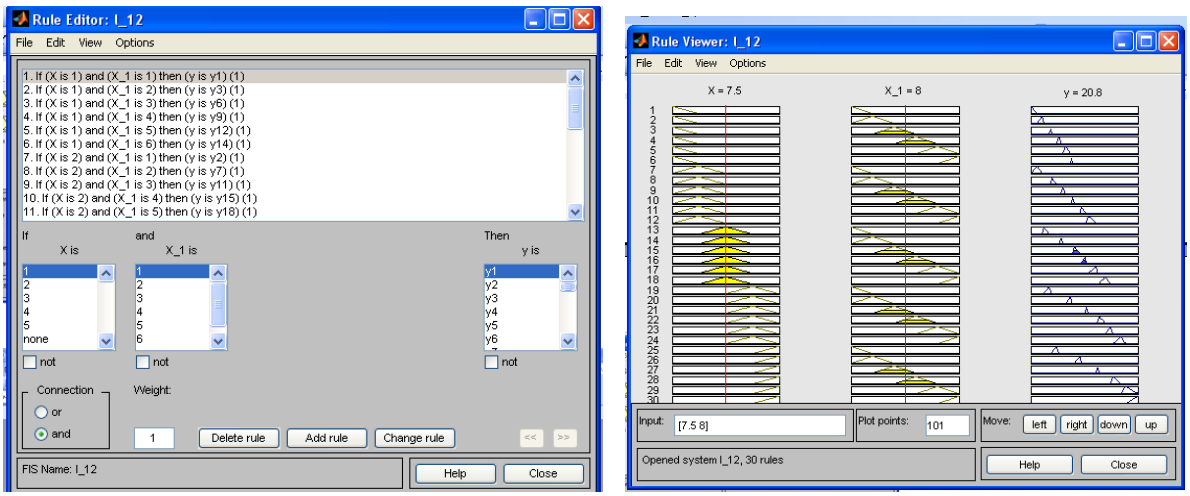
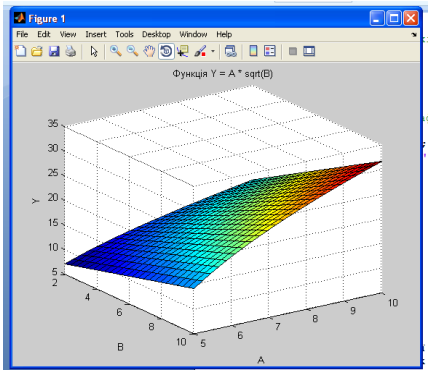
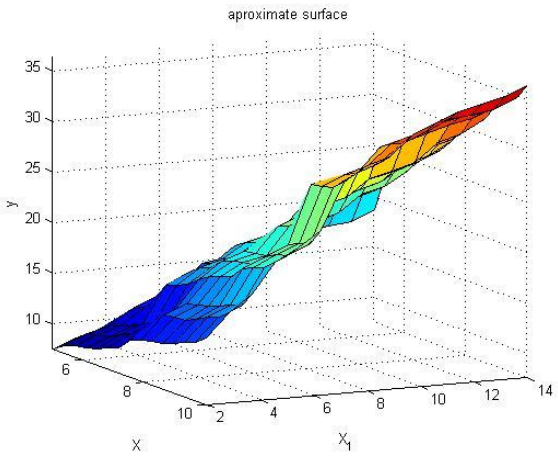


Рис. 3.14. Перегляд бази правил СНВ

Перегляд поверхні системи нечіткого виведення

```
a = 4;a_beg = 5; a_end = 10;
b = 6;b_beg = 2;b_end = 10;
%Кількість термів A і B
n = a; m = b;
%Розмір масиву змінної A
nm=n*m; a_space=(a_end-a_beg)/(nm-1);
%Розмір масиву змінної B
b_space=(b_end-b_beg)/(nm-1);
%Формування вхідних чітких векторів A і B
A=a_beg:a_space:a_end; B=b_beg:b_space:b_end;
y=zeros(nm,nm);
%-------------------- Вихідні данні --------------------
for i=1:nm
  for j=1:nm
     Y(i,j)=A(i)*sqrt(B(j));
end;end
surf(A, B, Y); xlabel('A'); ylabel('B'); zlabel('Y');
title('Функція Y = A * sqrt(B)')
```





Тема 2. Нечітка модель Мамдані

4. Процес розробки нечіткої моделі Мамдані в режимі команд. Нечітка модель як універсальний апроксиматор

Мета роботи: навчитися: 1) розробляти систему нечіткого виведення (СНВ) Мамдані, здатну розв'язувати задачу апроксимації функції (залежності вхід-вихід); 2) моделювати нелінійні функції за допомогою СНВ Мамдані в середовищі MatLab (із та без використання пакету Fuzzy Logic Toolbox). *Об'єкт дослідження* – нечітка модель Мамдані на основі пакету Fuzzy Logic Toolbox.

Питання для опрацювання

1. Процес розробки СНВ Мамдані в режимі команд (приклади) [3].
2. Рекомендації з побудови бази правил для різних варіантів розташування опорних точок [7].

1. Завдання до практичної роботи

Постановка задачі про апроксимацію функції за допомогою системи нечіткого виведення. Розглянемо відображення $\mathbf{d} = f(\mathbf{x})$, де $\mathbf{x} = [x_1 \ldots x_N]^{\mathrm{T}}$ – вектор вхідних даних, $\mathbf{d} = [d_1 \ldots d_M]^{\mathrm{T}}$ – вектор вихідних даних. Вектор-функція $f(\cdot)$ вважається невідомою, але задано множину реалізацій функції f: $T = \{(x_1^i, \ldots, x_N^i, d_1^i, \ldots, d_M^i) : 1 \le i \le p;\ N \ge 1;\ M \ge 1\}$, де p– кількість реалізацій.

Необхідно побудувати СНВ, яка апроксимує функцію $f(\cdot)$ і описує перетворення системою вхідного сигналу у вихідний через функцію $F(\cdot)$. Критерієм якості нечіткого виведення є підбір такого значення кількості термів для вхідних лінгвістичних змінних, при якому середня абсолютна нев'язка ε, яка виникає внаслідок похибок апроксимації, задовольняє умову: $\varepsilon = \frac{1}{p}\sum_{i=1}^{p}\left|F(\mathbf{x}^i) - \mathbf{d}^i\right| \le eps$, $\forall eps \ge 0$.

Алгоритм нечіткого виведення Мамдані:

Крок 1. Оцінка ступеня виконання умови.

Крок 2. Визначення активізованих функцій належності висновків окремих правил при заданих входах нечіткої моделі.

Крок 3. Визначення результуючої функції належності.

Крок 4. Дефазифікація (приведення до чіткості)

Завдання. 1. Розглянути і засвоїти процес розробки СНВ Мамдані в системі MatLab в інтерактивному режимі.

2. Розглянути і засвоїти процес розробки СНВ в середовищі MatLab в режимі команд: 1) написати програму, яка реалізує нечітку модель Мамдані, що апроксимує задану аналітичну залежність у визначеному діапазоні змінних, 2) визначити метод дефаззифікації, застосування якого забезпечить найкращу якість нечіткого виведення, обчислити нев'язку, 3) зробити висновки та оформити звіт.

2. Контрольні запитання

1. Намалюйте структуру нечіткої моделі (або СНВ) Мамдані. Опишіть її основні модулі. Опишіть алгоритм нечіткого виведення Мамдані. Наведіть приклад роботи даного алгоритму на базі знань, яка складається із двох правил.

2. Модуль «*Виведення»* нечіткої моделі Мамдані*:* опишіть його елементи та принцип функціонування.

3. Агрегування (оцінка ступеня виконання складних умов, зображених у вигляді комбінації простих): опишіть як ця операція реалізована в системах нечіткого виведення.

4. Виведення на правилах з використанням операторів нечіткої імплікації: опишіть процес визначення активізованих функцій належності висновків окремих правил при заданих вхідних значеннях нечіткої моделі.

5. Опишіть алгоритм виведення в СНВ типу Мамдані.

6. Параметри нечіткої моделі*:* оператори t-норми, s-норми (визначення, приклади).

7. Параметри нечіткої моделі*:* оператор імплікації (види, приклади).

8. Основні методи дефазифікації: формули обчислення, переваги і недоліки.

Задачі для самостійного розв'язання: *моделювання відношення вхід-вихід, заданого на основі функції від двох змінних*

Варіанти	Аналітична залежність	Кількість		Діапазони	
		a	*b*	*a*	*b*
1	$y = a\sqrt{b}$	5	4	[2; 4]	[2; 6]
2	$y = a \cdot b^3$	4	6	[10; 14]	[1; 5]
3	$y = \frac{a + ab - \sqrt{b}}{a + b}$	4	5	[1; 5]	[0,5; 2]
4	$y = \frac{a + b}{\sqrt{a + b^2 - 1}}$	6	5	[1; 3]	[1; 5]
5	$y = \frac{a^3 \sin(0.2b)}{a + b^2}$	4	5	[0,5; 3]	[0,5; 5]
6	$y = b(\sin(\frac{\pi}{6}a))^2$	4	5	[1; 6]	[0,5; 3]
7	$y = b\sin\left(a\frac{\pi}{2}\right)$	5	4	[1; 4]	[1; 4]
8	$y = \frac{a^2 - b^3}{2a + b}$	5	4	[1; 5]	[2; 6]
9	$y = \frac{9a - \sqrt{b}}{\sqrt{a} + b}$	2	3	[0,1; 1]	[0,05; 2]
10	$y = \frac{a - b^2}{ab}$	4	6	[0,5; 2,5]	[1; 4]

Варіанти	Аналітична залежність	Кількість термів		Діапазони	
		a	*b*	*a*	*b*
11	$y=(a-4)(b-5)^3$	3	5	[20; 25]	[2; 7]
12	$y=7\sqrt{a}-6ab$	6	4	[1; 2]	[10; 12]
13	$y=a^3\sin(\frac{\pi}{3}b)-\sqrt{b}$	6	4	[1; 6]	[5; 7]
14	$y=a^2-b^3\sin((\pi/3)\cdot a)$	6	3	[-2; 3]	[1; 5]
15	$y=\frac{a+b^2}{2\sqrt{ab}}$	4	5	[1; 7]	[1; 7]
16	$y=0.01\,a^2\,\mathrm{tg}(b)$	6	4	[-5; 5]	[-1; 1,5]
17	$y=b\cos\left(a\frac{\pi}{2}\right)$	5	3	[0,1; 3]	[0,5; 2,5]
18	$y=(a+b)cos(b)$	3	4	[1; 7]	[1; 7]
19	$y=(1+\cdot sin(a)^2)^b$	4	5	[1; 3]	[0,5; 2]
20	$y=(a^2-b^2)/(2a+b)$	3	4	[1; 4]	[3; 6]
21	$y=2ab^3$	4	6	[10; 15]	[1; 7]
22	$y=(\sin(a)\sin(b))/(ab)$	4	5	[1; 10]	[1; 10]
23	$y=\sqrt[3]{a+b}$	4	5	[5; 10]	[5; 10]

Аудиторна робота

Приклад 4.1 [3, с. 65– 69]. Процес розробки системи нечіткого виведення в режимі команд

1. Побудувати СНВ типу Мамдані, яка моделює задану аналітичну залежність у визначеному діапазоні.

2. Визначити метод дефаззифікації, застосування якого забезпечить найкращу якість нечіткого виведення.

Алгоритм виконання завдання. Розв’язати задачу апроксимації функції $y=x^2$, де $x\in[0{,}1;\ 2{,}1]$, кількість термів для вхідної змінної *x* дорівнює «6».

1. Розробка СНВ типу Мамдані, яка моделює задану аналітичну залежність

А. *Фазифікація вхідних і вихідних змінних*. В якості терм-множини вхідної змінної X будемо використовувати множину {*x*1, *x*2, *x*3, *x*4, *x*5, *x*6} з симетричними трикутними функціями належності, зображенними на рис. 4.1, а в якості терм-множини вихідної змінної Y будемо використовувати множину {*y*1, *y*2, *y*3, *y*4, *y*5, *y*6} з симетричними трикутними функціями належності, зображенними на рис. 4.2.

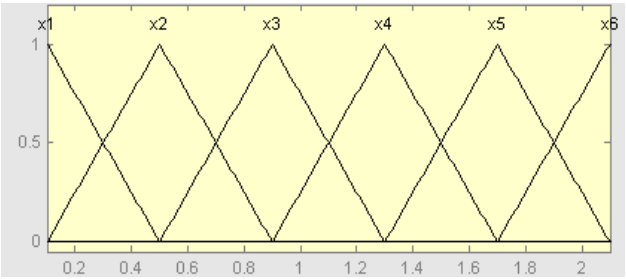


Рис. 4.1. Графіки ФН для термів вхідної змінної X

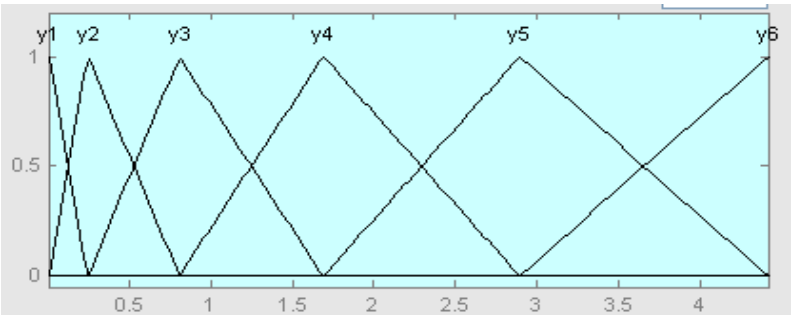


Рис.4.2. Графіки ФН для термів вихідної змінної Y

Б. *Формування нечіткої асоціативної матриці* (табл. 4.1), яка описує відношення вхід–вихід.

Таблиця 4.1

Нечітка асоціативна матриця

Y **X**	**0.01≤y<0.25**	**0.01≤y<0.81**	**0.25≤y<1.69**	**0.81≤y<2.89**	**169≤y<4.41**	**2.89≤y<4.41**
0.1≤x<0.5	*y*1					
0.1≤x<0.9		*y*2				
0.5≤x<1.3			*y*3			
0.9≤x<1.7				*y*4		
1.3≤x<2.1					*y*5	
1.7≤x<2.1						*y*6

В.*Формування системи нечіткого виведення*

```
function [X,Y,outt,ai,eps]=lab_rob_3(f_name,x_t,x_beg,x_end)
% Формування системи нечіткого виведення (типу Мамдані)
% Вхідні параметри:f_name–імя файлу; кількість термів x_t=y_t;
% діапазон можливих значень вхідної змінної x - [x_beg, x_end]
% Формування масиву значень вхідної змінної X
n=x_t; X=x_beg:((x_end-x_beg)/(n-1)):x_end;
delta_x=(x_end-x_beg)/(x_t-1)
% Формування масиву та діапазону значень вихідної змінної Y
for i=1:n
  Y(i)=X(i)^2;
end;
```

```
ymin=Y(1); ymax=Y(1);
for i=1:n
  if Y(i)<ymin
   ymin=Y(i)
  end
  if Y(i)>ymax
   ymax=Y(i)
  end
end; delta_y=(ymax-ymin)/x_t;
% Побудова графіку функції
xlabel('X');ylabel('Y');
axis ([x_beg-0.5 x_end+0.5 ymin-0.5 ymax+0.5]);
plot (X, Y, '+r')
% Побудова системи нечіткого виведення
ai=newfis(f_name,'mamdani','min','max','min','max','centroid');
ai=addvar(ai,'input','x',[x_beg x_end]);
for i=1:n
  if i==1
    a=x_beg-delta_x; else a=X(i-1)
  end
  b=X(i);
  if i==n c=x_end+delta_x; else c=X(i+1)
  end
  ai=addmf(ai,'input',1, ['x',int2str(i)],'trimf', [a b c]);
end
ai=addvar(ai,'output','y', [ymin ymax]);
for i=1:n
  if i==1
    a=ymin-delta_y; else a=Y(i-1)
  end
  b=Y(i);
  if i==n
    c=ymax+delta_y; else c=Y(i+1)
  end
  ai=addmf(ai,'output',1,['y',int2str(i)],'trimf',[a b c]);
end
% Формування бази знань системи
  rullist=ones(n, 4);
  for i=1:n
   rullist(i,1)=i;rullist(i,2)=i;
  end
ai=addrule(ai,rullist);
% Збереження СНВ у файлі поточного каталогу
 writefis(ai, f_name);
% Обчислення середньої абсолютної нев'язки
 ai=readfis(f_name);
 ai=setfis(ai,'defuzzmethod','centroid');
 outt=evalfis(X', ai);
 eps=sum(abs(outt-Y'))/n %обчислення середньої абс. нев'язки
```

Виклик m-функції, збереженої під ім'ям *l_rob_3.m*:

```
[X,Y,outt,ai,eps]=l_rob_2('l_rob_2', 6, 0.1, 2.1)
```

2. *Перегляд одержаного файлу l_rob_*3.*fis*:

а) Переглянути одержаний файл в середовищі складових пакету Fuzzy Logic Toolbox: редактора CHB FIS; редактора функцій належності; редактора бази знань; програми перегляду правил CHB.

б) Дослідити механізм нечіткого логічного виведення за допомогою команди Rules меню View.

3. *Обчислення середньої абсолютної нев'язки для різних методів дефаззифікації*:

А. Обчислити середню абсолютну нев'язку для таких методів дефаззифікації: 1) центра ваги для дискретної множини значень функції належності (*centroid*); 2) центра площини (*bisector*); 3) лівого модального значення (*som*); 4) правого модального значення (*lom*); 5) середнього максимуму, який визначається як середнє арифметичене лівого та правого модальних значень і видає коректний результат тільки у випадку унімодальної нечіткої множини (*mom*).

Б. Систематизувати одержані дані у формі таблиці і визначити який із наведених методів є найбільш ефективним.

В. Зробити висновки.

4. *Обчислення нев'язки для різних значень кількості термів*:

З'ясувати при якому значенні кількості термів для вхідних змінних ми одержимо наперед задану точність апроксимації (0,01 або 0,1).

Приклад 4.2. Моделювання відношення вхід-вихід, заданого на основі функції від двох змінних $y=f(x_1, x_2)$

1. Для функції, заданої у вигляді прикладу 3.1.

```
script % function lab_2_2var
clc;close all;clear all;
f_name='lab_21'; title='logic function'
% const-----------------------
Na=2; a0=0;an=1;
Nb=2;b0=0;bn=1;
% prepare-----------------------
da=(an-a0)/(Na-1);
db=(bn-b0)/(Nb-1);
A=a0:da:an; B=b0:db:bn;
[a,b]=meshgrid(A,B);
y=[1 1 1 0];
% figure(1);surf(a,b,y);xlabel('A'); ylabel('B'); zlabel('Y')
Ny=2;
y0=min(min(y)); yn=max(max(y)); dy=(yn-y0)/(Ny-1);
Y=y; y=y(:);y=sort(y);
% initialization-------------------
ai=newfis('f_name', 'mamdani', 'min', 'max','min', 'max',
'centroid');
ai= addvar(ai,'input','a', [a0 an]);
for i=1:Na
  if i==1
```

```
    a=a0-da;
  else
    a=A(i-1);
  end;
  b=A(i);
  if i==Na
    c=an+da;
  else
    c=A(i+1);
  end
  ai=addmf(ai,'input',1, ['a',int2str(i)],'trimf', [a b c]);
end;
ai=addvar(ai,'input','b', [b0 bn]);
for i=1:Nb
  if i==1
    a=b0-db;
  else
    a=B(i-1);
  end;
  b=B(i);
  if i==Nb
    c=bn+db;
  else;
    c=B(i+1);
  end
  ai=addmf(ai,'input',2, ['2',int2str(i)],'trimf', [a b c]);
end;
ai= addvar(ai,'output','y', [y0 yn]);
for i=1:Ny
  if i==1
    a=y0-dy;
  else
    a=y(i-1)
  end;
   b=y(i);
  if i==Ny;
   c=yn+dy;
  else
   c=y(i+1)
  end
  ai=addmf (ai,'output',1, ['y',int2str(i)],'trimf', [a b c]);
end;
%rulelist-------------------------------------
rullist=ones(4,5); s=0;
rullist(2,:)=[1 2 1 1 1];
rullist(3,:)=[2 2 1 1 1];
rullist(4,:)=[2 2 2 1 1];
ai= addrule(ai, rullist);
writefis (ai, 'f_name'); ai=readfis('f_name');
% Result-------------------------------------
x=[0.1 0.1]; a_check=0.1; b_check=0.1;Y_check=0.332;
```

```
%обчислення нев'язки моделювання функції СНВ при використанні
defuzzmethod
ai=setfis(ai,'defuzzmethod','centroid');
% fuzzy (ai)
out=evalfis([a_check b_check], ai); ep=abs(out-Y_check)
```

ep = 0.0082

2. Для функції вигляду $y=f(x_1, x_2)=x_1 x_2^2$

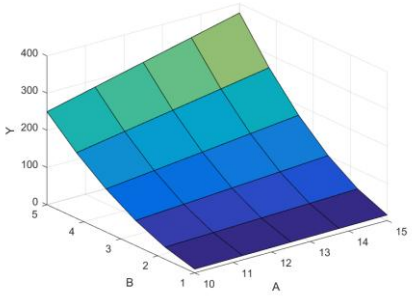
```
script % function lab_2_2var
clc;close all;clear all;
f_name='lab_21'; title='a.*(b.^2))'
% const-----------------------
Na=5; % кількість термів a
a0=10;an=15;
Nb=6;b0=1;bn=5;
% prepare----------------------
da=(an-a0)/(Na-1); db=(bn-b0)/(Nb-1);
A=a0:da:an; B=b0:db:bn;
[a,b]=meshgrid(A,B);y=a.*(b.^2);
% figure(1);surf(a,b,y);xlabel('A'); ylabel('B'); zlabel('Y')
Ny=Na*Nb;
y0=min(min(y)); yn=max(max(y)); dy=(yn-y0)/(Ny-1);
Y=y; y=y(:);y=sort(y);
% initialization------------------
ai=newfis('f_name', 'mamdani', 'min', 'max','min', 'max',
'centroid');
% adding var a
ai= addvar(ai,'input','a', [a0 an]);
for i=1:Na
  if i==1
    a=a0-da;
  else
    a=A(i-1);
  end;
  b=A(i);
  if i==Na
    c=an+da;
  else
    c=A(i+1);
  end
  ai=addmf(ai,'input',1, ['a',int2str(i)],'trimf', [a b c]);
end;
% adding var b --------
ai=addvar(ai,'input','b', [b0 bn]);
for i=1:Nb
  if i==1
    a=b0-db;
  else
    a=B(i-1);
  end;
```

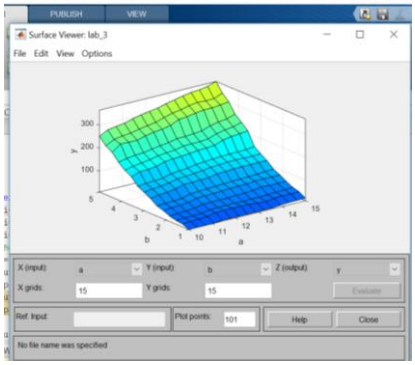
```
  b=B(i);
  if i==Nb
    c=bn+db;
  else;
    c=B(i+1);
  end
  ai=addmf(ai,'input',2, ['2',int2str(i)],'trimf', [a b c]);
end;
% adding var y
ai= addvar(ai,'output','y', [y0 yn]);
for i=1:Ny
  if i==1
    a=y0-dy;
  else
    a=y(i-1)
  end;
   b=y(i);
  if i==Ny;
   c=yn+dy;
  else
   c=y(i+1)
  end
  ai=addmf (ai,'output',1, ['y',int2str(i)],'trimf', [a b c]);
end;
%rulelist-------------------------------------
rullist=ones(Ny,5); s=0;
 for i=1:Na
   for j=1:Nb
       k=s+j; rullist(k,1)=i;rullist(k,2)=j;
       rullist(k,3)=find(y==Y(j,i));
   end;
   s=s+j;
 end;
ai= addrule(ai, rullist); writefis (ai, 'f_name');
ai=readfis('f_name');
%Result-----------------------------------
a_check=11:0.2:13; b_check=4; rr=length(a_check);
Y_check(1:rr)=a_check(1:rr)*(b_check)^2;
%обчислення нев'язки моделювання функції СНВ при використанні
різних defuzzmethod
ep=0; ai=setfis(ai,'defuzzmethod','centroid');
for i = 1:rr
   out(i)=evalfis([a_check(i) b_check], ai);
   ep=ep+abs(out(i)-Y_check(i));
end
ep=ep/rr; %fuzzy 'lab_31'
```

Результат

```
Y_check= [176.0000  179.2000  182.4000  185.6000  188.8000
192.0000  195.2000  198.4000  201.6000  204.8000  208.0000]
```

```
Out=[175.5810  181.0615  184.7536  186.8921  189.5487  191.5496
193.2622  198.8661  202.0677   201.4717  202.6045]
error = 1.7019
```





Приклад 4.3. Моделювання відношення вхід-вихід, заданого на основі множини пар «вхід – вихід (значення функції)»:

$$\left\{\mathbf{p}_1=\begin{bmatrix}0\\0\end{bmatrix},\ t_1=0\right\},\ \left\{\mathbf{p}_2=\begin{bmatrix}0\\1\end{bmatrix},\ t_2=0\right\},\ \left\{\mathbf{p}_3=\begin{bmatrix}1\\0\end{bmatrix},\ t_3=0\right\},\ \left\{\mathbf{p}_4=\begin{bmatrix}1\\1\end{bmatrix},\ t_4=1\right\}.$$

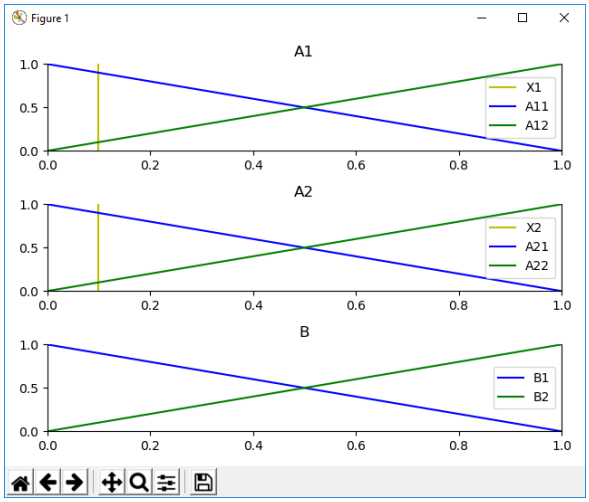
Лістинг:

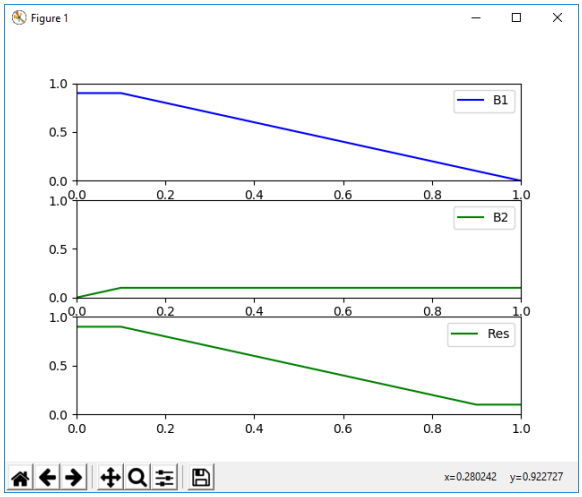
```
import numpy as np
import skfuzzy as fuzz
import matplotlib.pyplot as plt

x = [0.1, 0.1]
A = np.arange(0, 1.1, 0.1)
B = np.arange(0, 1.1, 0.1)
vert0 = A.copy()
vert1 = A.copy()
for i in range(len(vert0)):
    vert0[i] = x[0]
    vert1[i] = x[1]

A11=fuzz.trimf(A, [0, 0, 1]) # побудова трикутника нечіткої змінної
A12 = fuzz.trimf(A, [0, 1, 1])
B1 = fuzz.trimf(A, [0, 0, 1])
B2 = fuzz.trimf(A, [0, 1, 1])
# Visualize these universes and membership functions
fig, (ax0, ax1, ax2) = plt.subplots(nrows=3)
ax0.set_xlim([0,1])
ax0.set_ylim([0,1])
ax1.set_xlim([0,1])
ax1.set_ylim([0,1])
ax2.set_xlim([0,1])
ax2.set_ylim([0,1])

ax0.plot(vert0, A11, 'y', linewidth=1.5, label='X1')
ax0.plot(A, A11, 'b', linewidth=1.5, label='A11')
ax0.plot(A, A12, 'g', linewidth=1.5, label='A12')
ax0.set_title('A1')
```

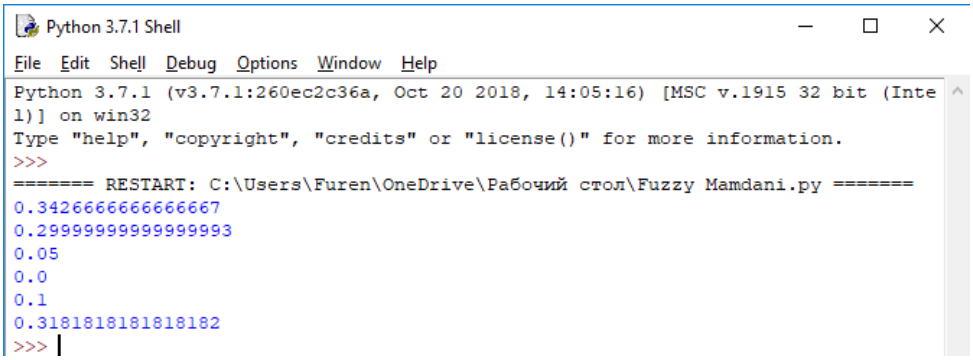
```
ax0.legend()
ax1.plot(vert1, A11, 'y', linewidth=1.5, label='X2')
ax1.plot(A, A11, 'b', linewidth=1.5, label='A21')
ax1.plot(A, A12, 'g', linewidth=1.5, label='A22')
ax1.set_title('A2')
ax1.legend()

ax2.plot(B, B1, 'b', linewidth=1.5, label='B1')
ax2.plot(B, B2, 'g', linewidth=1.5, label='B2')
ax2.set_title('B')
ax2.legend()

# Turn off top/right axes
for ax in (ax0, ax1, ax2):
    ax.spines['top'].set_visible(False)
    ax.get_xaxis().tick_bottom()
    ax.get_yaxis().tick_left()

plt.tight_layout()
plt.show()
a11_level = fuzz.interp_membership(A, A11, x[0])
a12_level = fuzz.interp_membership(A, A12, x[0])
a21_level = fuzz.interp_membership(A, A11, x[1])
a22_level = fuzz.interp_membership(A, A12, x[1])
rullist = []
rullist = [[0,0,0],
           [0,1,0],
           [1,0,0],
           [1,1,1]]
r = [0,0,0,0]
key = 0; act = [0,0]
for i in rullist:
    if i[0] == 0:
        v1 = a11_level
    else:
        v1 = a12_level
    if i[1] == 0:
        v2 = a21_level
    else:
        v2 = a22_level
    r[key] = np.fmin(v1,v2)
    key+=1
for i in range(4):
    j = rullist[i][2]
    act[j] = np.fmax(act[j],r[i])
res = B1.copy()
for i in range(len(B1)):
    if B1[i]>act[0]: B1[i] = act[0]
    if B2[i]>act[1]: B2[i] = act[1]
    res = np.fmax(B1,B2)
fig, (ax0, ax1, ax2) = plt.subplots(nrows=3)
ax0.set_xlim([0,1])
```

```
ax0.set_ylim([0,1])
ax1.set_xlim([0,1])
ax1.set_ylim([0,1])
ax2.set_xlim([0,1])
ax2.set_ylim([0,1])
ax0.plot(B, B1, 'b', linewidth=1.5, label='B1')
ax1.plot(B, B2, 'g', linewidth=1.5, label='B2')
ax2.plot(B, res, 'g', linewidth=1.5, label='Res')
ax0.legend();ax1.legend();ax2.legend()
plt.show()
s = 0
for i in range(len(B)):
    s += B[i]*res[i]
for metod in ['centroid',
              'bisector',
              'mom',
              'som',
              'lom']:
    y = fuzz.defuzz(B,res,metod)
    print(y)
s1 = s2 = 0
for i in range(len(B)):
    s1+=B[i]*res[i]; s2+=res[i]
print(s1/s2)
```

Результат







Теоретичні відомості

1. Рекомендації з побудови бази правил для різних варіантів розташування опорних точок [7]. База правил повинна забезпечувати можливість досягнення необхідної точності нечіткої моделі (після того, як визначені параметри останньої). Одночасно з цим, щоб зробити модель більш зрозумілою, кількість правил, які містяться в базі, має бути якомога меншою (скорочення кількості правил в моделі з кількома входами може бути попередньою вимогою для виконання налаштування її параметрів).

Зазначені властивості нечіткої моделі – точність і кількість правил – є взаємовиключними: зменшення кількості правил в моделі у загальному випадку знижує її точність. При виборі кількості правил необхідно враховувати такі рекомендації:

1. Кількість правил збільшується при ущільненні сітки, яку використовують для розбиття простору X входів моделі.

2. Щільність сітки, яку використовують для розбиття, слід збільшувати у випадку більш рельєфної поверхні відображення $X \rightarrow Y$ моделі. У випадку плоскої (близькою до лінійної) поверхні необхідність у такому розбитті відсутня

3. При незмінній щільності сітки (незмінній кількості правил) точність моделі може бути підвищена шляхом правильного розміщення опорних точок, які задаються правилами її поверхні.

Після того, як обрана щільність сітки розбиття, можна приступати до формування правил, які задають опорні точки поверхні моделі. Існує два методи розташування опорних точок: *метод* 1 - розташування точок по кутах прямокутних сегментів сітки розбиття; *метод* 2 - розташування точок у центрі сегментів. На рис. 4.3 наведено приклад використання методу 1.

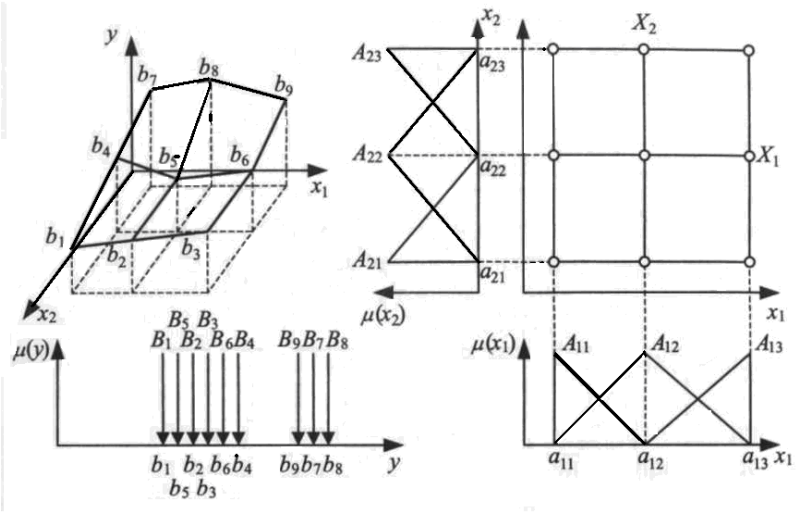


Рис. 4.3. Приклад нечіткої моделі з розташуванням опорних точок по кутах сегментів розбиття простору $X=X_1\times X_2$ вхідних значень

***Метод* 1**. Якщо правила визначаються для кутових точок прямокутних сегментів простору вхідних значень, то відповідна кожному сегменту ділянка поверхні моделі задається чотирма правилами, які відповідають його кутовим точкам. У випадку моделі (рис. 4.3) для визначення поверхні, що відповідає всій області значень $X=X_1\times X_2$, використовують дев'ять правил вигляду:

R1: ЯКЩО (x_1=A_{11}) І (x_2=A_{21}) ТО (y=B_1)
R2: ЯКЩО (x_1=A_{12}) І (x_2=A_{21}) ТО (y=B_2)
R3: ЯКЩО (x_1=A_{13}) І (x_2=A_{21}) ТО (y=B_3)
R4: ЯКЩО (x_1=A_{11}) І (x_2=A_{22}) ТО (y=B_4)
R5: ЯКЩО (x_1=A_{12}) І (x_2=A_{22}) ТО (y=B_5)
R6: ЯКЩО (x_1=A_{13}) І (x_2=A_{22}) ТО (y=B_6)
R7: ЯКЩО (x_1=A_{11}) І (x_2=A_{23}) ТО (y=B_7)
R8: ЯКЩО (x_1=A_{12}) І (x_2=A_{23}) ТО (y=B_8)
R9: ЯКЩО (x_1=A_{13}) І (x_2=A_{23}) ТО (y=B_9)

Наведені правила містять інформацію про вихідні значеннях моделі при вхідних станах (x_1, x_2), які в точності відповідають кутовим точкам сегментів. Наприклад, для $x_1=a_{12}$, $x_2=a_{22}$ вихідний стан має вигляд $y=b_5$. У просторі між кутовими точками нечітка модель виконує інтерполяцію, характер якої залежить від методів виведення і дефазифікації, а також типу ФН.

Метод 2. В рамках нього опорні точки, що задаються правилами, розміщуються в центрі сегментів (рис. 4.4). Використовуючи метод 2 для тієї ж області значень X, отримуємо базу правил вигляду:

*R*1: ЯКЩО (x_1=A_{11}) І (x_2=A_{21}) ТО *(y=B_3)*
R2: ЯКЩО (x_1=A_{11}) І (x_2=A_{22}) ТО *(y=B_1)*
R3: ЯКЩО (x_1=A_{12}) І (x_2=A_{21}) ТО *(y=B_4)*
R4: ЯКЩО (x_1=*A_{12})* І (x_2=A_{22}) ТО *(y=B_2)*

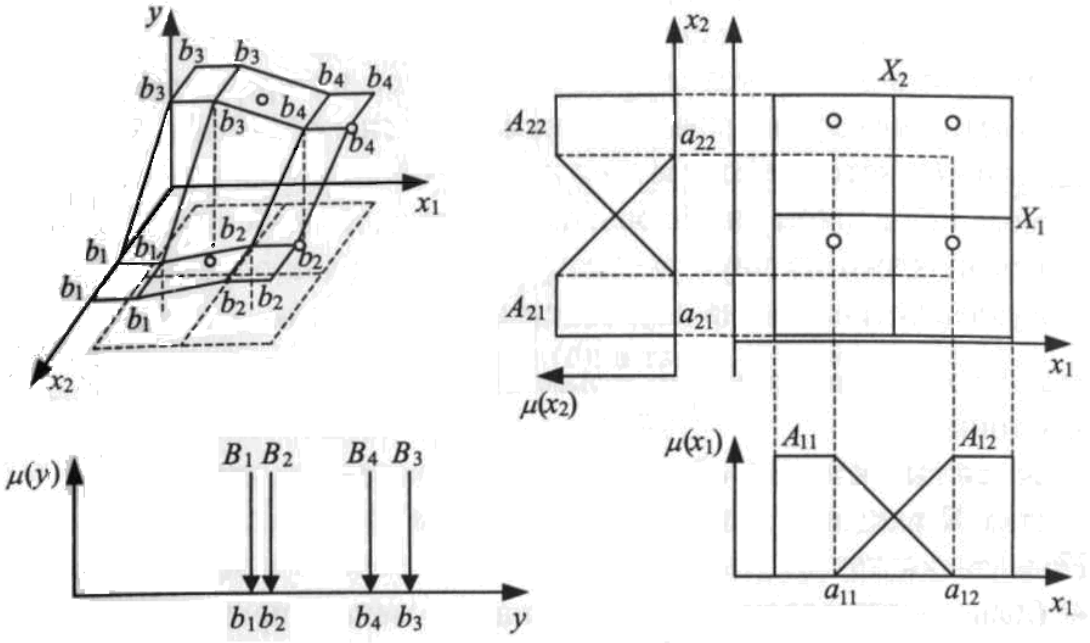


Рис. 4.4. Приклад нечіткої моделі, розміщення опорних точок якої задається правилами у центрі сегментів розбиття області значень X=X_1×X_2

Правила (7.60) містять точну інформацію про вихідні стани моделі в точках, які відповідають (більшою чи меншою мірою) центрам сегментів. Для точок, що знаходяться між ними, вихідне значення моделі обчислюється на основі нечіткої інтерполяції. За межами ділянки між опорними точками помітні області насичення, зі значеннями b_i, які відповідають найближчим опорним точкам. Точність моделі в даному діапазоні, як правило, є низькою.

Можна вказати на такі переваги та недоліки кожного з двох розглянутих методів визначення правил:

Метод 1 (опорні точки, визначені правилами, розміщуються в кутах прямокутних сегментів сітки розбиття). *Переваги*: більш висока точність моделі, в тому числі на кордонах простору X вхідних значень. *Недоліки*:

1) більше число правил, що приводить до менш «прозорих» (інтуїтивно зрозумілих) моделейм, 2) для визначення правил необхідний більший обсяг інформації.

Метод 2 (опорні точки, визначені правилами, розміщуються в центрі сегментів сітки розбиття). *Переваги*: 1) менша, в порівнянні з методом 1, кількість правил, які призводять до більшої «прозорості» моделі, 2) менший обсяг вимірюваної інформації, необхідної для формування правил. *Недоліки*: менша точність моделі в порівнянні з методом 1, особливо на кордонах області значень.

Перед початком налаштування моделі параметри опорних точок a_{ij}, b_k можуть, наприклад, бути розподілені рівномірно. Забезпечуючи більш високу точність моделі, у процесі налаштування відбувається зміна позицій опорних точок (параметрів нечітких множин в правилах). Більша кількість опорних точок може привести до досягнення більшої точності нечіткої моделі (за умови ефективно виконаного налаштування), але одночасно з цим процес налаштування моделі стає більш складним.

5. Нечітка модель Мамдані та функції користувача

Мета роботи: ознайомитися з процедурою створення функцій користувача для СНВ Мамдані. *Об'єкт дослідження* – нечітка модель Мамдані на основі пакету Fuzzy Logic Toolbox, функції користувача для СНВ.

Питання для опрацювання

1. Нечіткі оператори [4, 7].
2. Нечітка імплікація [4, 7].

1. Завдання до практичної роботи

Постановка задачі. Розглянемо відображення $y=f(x_1,\ldots,x_n)$. Визначимо початкові (експериментальні) дані таким чином: множина вхідних параметрів $X=\{x_i, i=1,\ldots,n\}$, де x_i – змінні, які приймають числові значення, тобто $x_i \in \left[\underline{x}_i, \overline{x}_i\right]$; вихідний параметр $y \in \left[\underline{y}_j, \overline{y}_j\right]$, $j=1,\ldots,m$.

Необхідно за заданим вектором значень вхідних параметрів $x^*=(x_1^*,\ldots,x_n^*)$ визначити розв'язок y на основі нечеткої моделі.

1. Створити функції користувача для заданих параметрів нечіткої моделі. Застосувати одержані функції користувача до практичної роботи № 3. Порівняти похибки одержаних моделей та визначити оптимальні параметри нечіткої моделі Мамдані.

2. Контрольні запитання

1. Визначити поняття «параметризовані та непараметризовані t-норми».
2. Визначити поняття «параметризовані та непараметризовані t-конорми»

Задачі для самостійного розв'язання: комп'ютерний практикум № 4

Застосувати

I. *Оператор об'єднання ФН змінної виходу*: $\mu_R(x,y)=\mathrm{MAX}(\mu_A(x),\mu_B(y))$;

II. *Оператор дефазифікації*: метод центра ваги (Center of Gravity, CG),

Створити функції користувача для заданих *t*-норми та імплікацій

№	*t-норма*	*нечіткі імплікації*
1	min	Мамдані; Лукасевича
2	добуток (алгебраїчний перетин)	Добуток; Лукасевича
3	добуток Гамахера	Мамдані; Заде
4	добуток Ейнштейна	Добуток; Заде
5	Лукасевича	Добуток; Гогена
6	min	Мамдані; Гьоделя
7	добуток (алгебраїчний перетин)	Добуток; Гьоделя
8	добуток Гамахера	Мамдані; Кліні-Дінса
№	*t-норма*	*нечітка імплікація:*
9	добуток Ейнштейна	Добуток; Кліні-Дінса
10	Лукасевича	Гогена; Кліні-Дінса
11	min	Мамдані; Кліні-Дінса- Лукасевича
12	добуток (алгебраїчний перетин)	Добуток; Кліні-Дінса- Лукасевича

13	добуток Гамахера	Мамдані; Лукасевича
14	добуток Ейнштейна	Добуток; Лукасевича
15	Лукасевича	Кліні-Дінса; Лукасевича
16	min	Мамдані; Заде
17	добуток (алгебраїчний перетин)	Добуток; Заде
18	добуток Гамахера	Мамдані; Гьоделя
19	добуток Ейнштейна	Добуток; Гьоделя
20	Лукасевича	Гьоделя; Кліні-Дінса-Лукасевича
21	min	Мамдані; Кліні-Дінса
22	добуток (алгебраїчний перетин)	Добуток; Кліні-Дінса
23	добуток Гамахера	Мамдані; Кліні-Дінса-Лукасевича
24	добуток Ейнштейна	Добуток; Кліні-Дінса-Лукасевича

Аудиторна робота

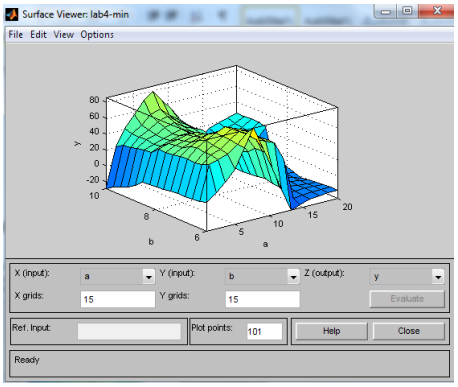
Приклад 5.1: Matlab. Застосування функцій користувача під час апроксимації заданих функцій

```
script % function lab_2var
clc;close all;clear all;
title='a.*(b.^2))'
% const-----------------------
Na=5; % кількість термів a
a0=10;an=15;
Nb=6;b0=1;bn=5;
% prepare-----------------------
da=(an-a0)/(Na-1);
db=(bn-b0)/(Nb-1);
A=a0:da:an; B=b0:db:bn;
[a,b]=meshgrid(A,B);y=a.*(b.^2);
%figure(1);surf(a,b,y);xlabel('A'); ylabel('B'); zlabel('Y')
Ny=Na*Nb; y0=min(min(y)); yn=max(max(y));
dy=(yn-y0)/(Ny-1); Y=y; y=y(:);y=sort(y);
% initialization------------------
funcs={'min','alg_int'}; %alg_int=prod
for f=1:length(funcs)
 rab=['l32' funcs{1}]
 ai=newfis('rab', 'mamdani', funcs{1}, 'max', 'min',
'max','centroid');
 % adding var a
 ai=addvar(ai,'input','a', [a0 an]);
 for i=1:Na
  if i==1

    a=a0-da;
  else
    a=A(i-1);
  end;
  b=A(i);
  if i==Na
    c=an+da;
```

```
  else
    c=A(i+1);
  end
  ai=addmf(ai,'input',1,['a',int2str(i)],'trimf',[a b c]);
 end;
% adding var b
 ai= addvar(ai,'input','b', [b0 bn]);
 for i=1:Nb
  if i==1
    a=b0-db;
  else
    a=B(i-1);
  end;
  b=B(i);
  if i==Nb
     c=bn+db;
  else
    c=B(i+1);
  end;
  ai=addmf(ai,'input',2, ['b',int2str(i)],'trimf', [a b c]);
 end;
 % adding var y
 ai= addvar(ai,'output','y', [y0 yn]);
 for i=1:Ny
  if i==1
     a=y0-dy;
  else
     a=y(i-1);
  end;
  b=y(i);
  if i==Ny;
    c=yn+dy;
  else
    c=y(i+1);
  end
  ai=addmf (ai,'output',1, ['y',int2str(i)],'trimf', [a b c]);
  end;
%rulelist-----------
rullist=ones(Ny,5); s=0;
 for i=1:Na
   for j=1:Nb
       k=s+j;rullist(k,1)=i; rullist(k,2)=j;
       rullist(k,3)=find(y==Y(j,i));
   end;
   s=s+j;
 end;
 ai= addrule(ai, rullist);
 writefis (ai, 'rab');
 ai=readfis('rab');
 %Result---------------------------------------
 a_check=11:0.2:13;b_check=4;
 rr=length(a_check);
```

```
 Y_check(1:rr)=a_check(1:rr)*(b_check)^2;
%обчислення нев'язки результату моделювання функції на основі СНВ
 ep=0; ai=setfis(ai,'defuzzmethod','centroid');
 for i = 1:rr
  out(i)=evalfis([a_check(i) b_check], ai);
  ep=ep+abs(out(i)-Y_check(i));
 end
 error(f)=ep/rr;
end;
error  %fuzzy 'lab_3'
```

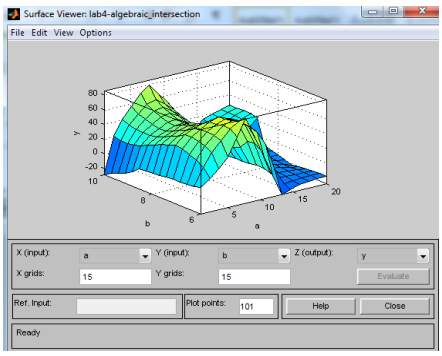
Результат апроксимації для функції користувача min



Algebraic_intersection

```
function U = algebraic_intersection(A, B)
 s=size(A);
 if (exist('B', 'var') ~= 1);
    if(s(1)==1 && s(2)~=1)
      A=A'; s=size(A);
    end
    str.type='()';
    str.subs=[1,num2cell(repmat(':',1, length(s)-1))];
    comp=subsref(A, str);
     for i=2:s(1)
      str.subs{1}=i;
          comp=algebraic_intersection(comp, subsref(A, str));
     end
     U = comp;
 else
     U = A.*B;
 end;
end
```

Результат апроксимації для функції користувача *алгебраїчний перетин*



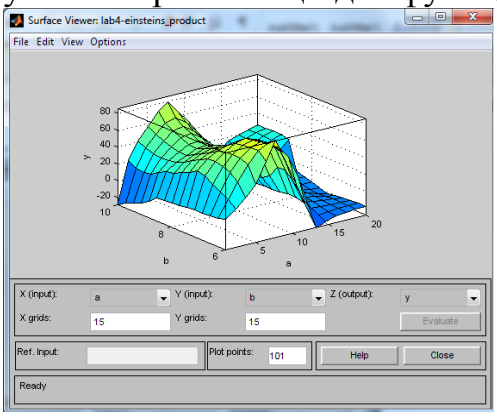
алгебраїчний перетин

$$\mu_{\bar{A}\wedge\bar{B}}(x)=\mu_{\bar{A}}(x)\times\mu_{\bar{B}}(x),$$

Einsteins_product

```
function U = einsteins_product(A, B)
 s=size(A);
 if(exist('B', 'var') ~= 1);
    if(s(1)==1 && s(2)~=1)
      A=A'; s=size(A);
    end
    str.type='()';
    str.subs=[1,num2cell(repmat(':',1, length(s)-1))];
    comp=subsref(A, str);
     for i=2:s(1)
      str.subs{1}=i;
          comp=einsteins_product(comp, subsref(A, str));
     end
     U = comp;
 else
     U = (A.*B)./(2-(A+B-(A.*B)));
 end; end
```

Результат апроксимації для функції користувача *добуток Ейнштейна*



добуток Ейнштейна

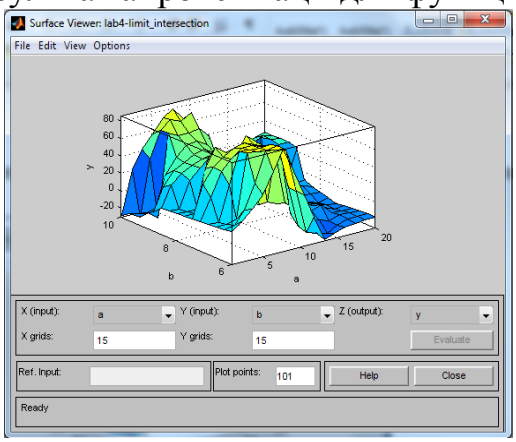
$$\mu_{\bar{A}\wedge\bar{B}}(x)=\frac{\mu_{\bar{A}}(x)\times\mu_{\bar{B}}(x)}{2-(\mu_{\bar{A}}(x)+\mu_{\bar{B}}(x)-\mu_{\bar{A}}(x)\times\mu_{\bar{B}}(x))},$$

Limit_intersection

```
function U = limit_intersection(A, B)
 s=size(A);
 if(exist('B', 'var') ~= 1);
    if(s(1)==1 && s(2)~=1)
      A=A'; s=size(A);
    end
    str.type='()';
```

```
    str.subs=[1,num2cell(repmat(':',1, length(s)-1))];
    comp=subsref(A, str);
     for i=2:s(1)
      str.subs{1}=i;
      comp=limit_intersection(comp, subsref(A, str));
     end
     U = comp;
 else
     U = max(zeros(s), (A+B-ones(s)));
 end; end
```

Результат апроксимації для функції користувача *граничний перетин*



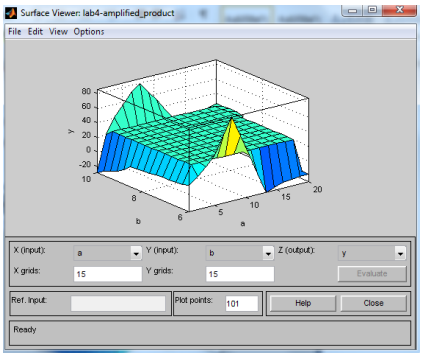
граничний перетин

$$\mu_{\bar{A}\wedge\bar{B}}(x) = max\{0, \mu_{\bar{A}}(x) + \mu_{\bar{B}}(x) - 1\}.$$

Amplified_product

```
function U = amplified_product(A, B)
 s=size(A);
 if(exist('B', 'var') ~= 1);
    if(s(1)==1 && s(2)~=1)
      A=A'; s=size(A);
    end
    str.type='()';
    str.subs=[1,num2cell(repmat(':',1, length(s)-1))];
    comp=subsref(A, str);
     for i=2:s(1)
      str.subs{1}=i;
          comp=amplified_product(comp, subsref(A, str));
     end
     U = comp;
 else
     U = min(A, B).*(max(A, B)==1);
 end; end
```

Результат апроксимації для функції користувача *посилений добуток*

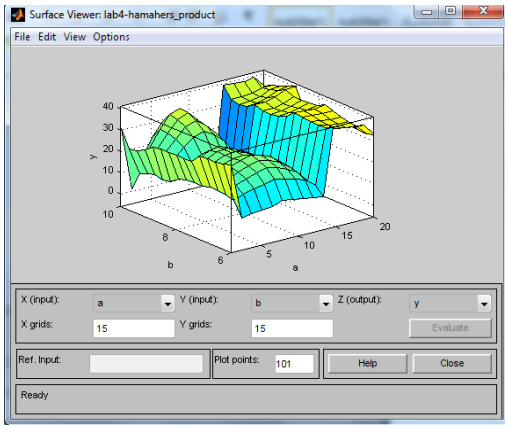


посилений добуток $\mu_{\bar{A}\wedge\bar{B}}(x)=\begin{cases} min(\mu_{\bar{A}}(x),\mu_{\bar{B}}(x)), якщо\ max\ (\mu_{\bar{A}}(x),\mu_{\bar{B}}(x))=1, \\ 0,\ інакше, \end{cases}$

Hamahers_product

```
function U = hamahers_product(A, B)
 s=size(A);
 if(exist('B', 'var') ~= 1);
    if(s(1)==1 && s(2)~=1)
      A=A'; s=size(A);
    end
    str.type='()';
    str.subs=[1,num2cell(repmat(':',1, length(s)-1))];
    comp=subsref(A, str);
     for i=2:s(1)
      str.subs{1}=i;
          comp=hamahers_product(comp, subsref(A, str));
     end
     U = comp;
 else
     U = (A.*B)./(A+B-(A.*B));
 end; end
```

Результат апроксимації для функції користувача *добуток Гамахера*



добуток Гамахера $\mu_{\bar{A}\wedge\bar{B}}(x)=\dfrac{\mu_{\bar{A}}(x)\times\mu_{\bar{B}}(x)}{\mu_{\bar{A}}(x)+\mu_{\bar{B}}(x)-\mu_{\bar{A}}(x)\times\mu_{\bar{B}}(x)}$

Maximum

```
function U = maximum (A, B)
 s=size(A);
 if(exist('B', 'var') ~= 1);
    if(s(1)==1 && s(2)~=1)
      A=A'; s=size(A);
    end
    str.type='()';
    str.subs=[1,num2cell(repmat(':',1, length(s)-1))];
    comp=subsref(A, str);
     for i=2:s(1)
      str.subs{1}=i;
          comp= maximum(comp, subsref(A, str));
     end
     U = comp;
 else
     U = max(A,B);
 end; end
```

Klini-dinsa-lukas

```
function U = klini_dinsa_lukas (A, B)
 s=size(A);
 if(exist('B', 'var') ~= 1);
    if(s(1)==1 && s(2)~=1)
      A=A'; s=size(A);
    end
    str.type='()';
    str.subs=[1,num2cell(repmat(':',1, length(s)-1))];
    comp=subsref(A, str);
     for i=2:s(1)
      str.subs{1}=i;
          comp= klini_dinsa_lukas (comp, subsref(A, str));
     end
     U = comp;
 else
     U = 1-(A+(A.*B));
 end; end
```

Теоретичні відомості

1. Нечіткі оператори [4, 17]. Допускається узагальнене зображення нечітких теоретико–множинних операцій через так звані нечіткі оператори. Ці оператори діють на множині значень функцій належності і тому їх можна застосовувати до функцій належності довільних нечітких множин. Найбільший інтерес серед нечітких операторів викликають трикутна норма й трикутна конорма.

Довільна дійсна функція від двох змінних T:[0, 1]×[0, 1]→[0, 1] називається *трикутною нормою* (*t–нормою*), якщо вона задовольняє такі умови (аксіоми трикутної норми):

1. Обмеженість: $T(x,0) = 0$, $T(x, 1) = x$;

2. Комутативність: $T(x, y) = T(y, x)$;

3. Асоціативність: $T(x,T(y, z)) = T(T(x, y), z)$;

4. Монотонність: $T(x, y) \le T(z_1, z_2)$, якщо $x \le z_1$ і $y \le z_2$.

Аксіома обмеженості забезпечує виконання граничних умов для всіх операцій перетину нечітких множин. Аксіоми комутативності та асоціативності забезпечують відповідно комутативність та асоціативність операцій перетину нечітких множин.

Розрізняють параметризовані й непараметризовані t–нормы. Результат дії непараметризованих t–норм є постійним, а для параметризованих t–норм він буде змінюватися при зміні будь–якого параметру. Серед найбільш поширених непараметризованих t–норм від двох аргументів (як один із основних параметрів нечіткої моделі – агрегації умов) можна виділити:

1) мінімум $\mu_{\bar{A}\cap\bar{B}}(x) = \min\{\mu_{\bar{A}}(x),\ \mu_{\bar{B}}(x)\}$,

2) алгебраїчний перетин (або алгебраїчний добуток)

$$\mu_{\bar{A}\cap\bar{B}}(x) = \mu_{\bar{A}}(x)\cdot\mu_{\bar{B}}(x),$$

3) Лукасевича: $\mu_{\bar{A}\cup\bar{B}}(x) = \max\left\{\mu_{\bar{A}}(x)+\mu_{\bar{B}}(x)-1,\ 0\right\}$

4) добуток Гамахера $\mu_{\bar{A}\cap\bar{B}}(x) = \dfrac{\mu_{\bar{A}}(x)\cdot\mu_{\bar{B}}(x)}{\mu_{\bar{A}}(x)+\mu_{\bar{B}}(x)-\mu_{\bar{A}}(x)\cdot\mu_{\bar{B}}(x)}$,

5) добуток Ейнштейна $\mu_{\bar{A}\cap\bar{B}}(x) = \dfrac{\mu_{\bar{A}}(x)\cdot\mu_{\bar{B}}(x)}{2-(\mu_{\bar{A}}(x)+\mu_{\bar{B}}(x)-\mu_{\bar{A}}(x)\cdot\mu_{\bar{B}}(x))}$,

6) посилений добуток

$$\mu_{\bar{A}\cap\bar{B}}(x) = \begin{cases} \min(\mu_{\bar{A}}(x),\mu_{\bar{B}}(x)), \text{якщо } \max\ (\mu_{\bar{A}}(x),\mu_{\bar{B}}(x)) = 1, \\ 0,\ \text{інакше}, \end{cases}$$

7) граничний перетин $\mu_{\bar{A}\cap\bar{B}}(x)=\max\{0,\ \mu_{\bar{A}}(x)+\mu_{\bar{B}}(x)-1\}$.

Параметризовані t–норми розглянуто в [6].

Довільна дійсна функція від двох змінних *S*: [0, 1]×[0, 1]→[0, 1] називається s–нормою або трикутною конормою (*t*–конормою), якщо вона задовольняє такі умови (аксіоми трикутної конорми):

1. Обмеженість: S(*x*, 0)=*x*; S(*x*, 1) = 1;

2. Комутативність: *S*(*x*, *y*) = *S*(*y*, *x*);

3. Асоціативність: *S*(*x*, *S*(*y*,*z*)) = *S*(*S*(*x*, *y*), *z*);

4. Якщо $x \le z_1$ і $y \le z_2$, то маємо монотонність: $S(x, y) \le S(z_1, z_2)$,.

Аксіома обмеженості конорми забезпечує виконання граничних умов для всіх операцій об'єднання нечітких множин.

Розрізняють параметризовані й непараметризовані t–конормы. Результат виконання непараметризованих операторів є постійним. Серед найбільш поширених непараметризованих *t*–конорм можна виділити:

1) максимум: $\mu_{\bar{A}\cup\bar{B}}(x)=\max\{\mu_{\bar{A}}(x),\mu_{\bar{B}}(x)\}$,

2) Лукасевича: $\mu_{\bar{A}\cup\bar{B}}(x)=\min\left\{\mu_{\bar{A}}(x)+\mu_{\bar{B}}(x),\ 1\right\}$,

3) алгебраїчна сума: $\mu_{\bar{A}\cup\bar{B}}(x)=\mu_{\bar{A}}(x)+\mu_{\bar{B}}(x)-\mu_{\bar{A}}(x)\mu_{\bar{B}}(x)$,

4) сумма Гамахера: $\mu_{\bar{A}\cup\bar{B}}(x)=\dfrac{\mu_{\bar{A}}(x)+\mu_{\bar{B}}(x)-2\mu_{\bar{A}}(x)\mu_{\bar{B}}(x)}{\mu_{\bar{A}}(x)+\mu_{\bar{B}}(x)-\mu_{\bar{A}}(x)\mu_{\bar{B}}(x)}$,

5) обмежена сума: $\mu_{\bar{A}\cup\bar{B}}(x)=\min(1,\ \mu_{\bar{A}}(x)+\mu_{\bar{B}}(x))$,

6) посилена сума (strong):

$$\mu_{\bar{A}\cup\bar{B}}(x)=\begin{cases}\max(\mu_{\bar{A}}(x),\mu_{\bar{B}}(x)), \text{якщо } \min(\mu_{\bar{A}}(x),\mu_{\bar{B}}(x))=0;\\ 1 \text{ інакше.}\end{cases}$$

Параметризовані t–конорми наведено в [6].

2. Нечітка імплікація [4, 17]. *Нечіткою імплікацією* висловлювань $\bar{A}$ і $\bar{B}$ ($\bar{A}\to\bar{B}$ читається «ЯКЩО $\bar{A}$, ТО $\bar{B}$») називають бінарну логічну операцію, результатом якої є нечітке висловлювання, значення істинності якого

визначається за обраною формулою. Найчастіше використовують такі формули:

1. Нечітка імплікація Заде:
$$T(\overline{A} \to \overline{B}) = \max\{\min\{T(\overline{A}), T(\overline{B})\}, 1 - T(\overline{A})\}.$$
2. Нечітка імплікація Гьоделя: $T(\overline{A} \to \overline{B}) = \begin{cases} 1, \text{якщо } T(\overline{A}) \le T(\overline{B}); \\ T(\overline{B}) \text{ інакше.} \end{cases}$
3. Нечітка імплікація Мамдані: $T(\overline{A} \to \overline{B}) = \min\{T(\overline{A}), T(\overline{B})\}$.
4. Нечітка імплікація Лукасевича:
$$T(\overline{A} \to \overline{B}) = \min\{1, T(\neg\overline{A}) + T(\overline{B})\}.$$
5. Нечітка імплікація Гогена (для $T(\overline{A}) > 0$):
$$T(\overline{A} \to \overline{B}) = \min\{1, T(\overline{B}) / T(\overline{A})\}.$$
6. Нечітка імплікація Ларсена
$$T(\overline{A} \to \overline{B}) = T(\overline{A})T(\overline{B}).$$
7. Нечітка імплікація Ягера: $T(\overline{A} \to \overline{B}) = T(\overline{B})^{T(\overline{A})}$.
8. Нечітка імплікація Кліні-Дінса
$$T(\overline{A} \to \overline{B}) = \max\{T(\neg\overline{A}), T(\overline{B})\}.$$
9. Нечітка імплікація Кліні-Дінса-Лукасевича
$$T(\overline{A} \to \overline{B}) = T(\neg\overline{A}) + T(\overline{A})T(\overline{B}).$$
10. Нечітка імплікація standart strict
$$T(\overline{A} \to \overline{B}) = \begin{cases} 1, \text{якщо } T(\overline{A}) \le T(\overline{B}); \\ 0 \text{ інакше.} \end{cases}$$
11. Нечітка імплікація Гайнса $T(\overline{A} \to \overline{B}) = \begin{cases} 1, \text{якщо } T(\overline{A}) \le T(\overline{B}); \\ T(\overline{B}) / T(\overline{A}) \text{ інакше.} \end{cases}$

6. Спрощення нечіткої моделі на основі скорочення Бази правил

Мета роботи: ознайомитися з методами спрощення нечіткої моделі на основі скорочення бази правил. *Об'єкт дослідження* – спрощена нечітка модель.

Питання для опрацювання

1. Скорочення бази правил [7].

1. Завдання до практичної роботи

1. Використовуючи программу формування нечіткої моделі, яка реалізує метод апроксимації функції (комп'ют. практикум 4), скоротити базу правил, видаляючи конкретні терми. Визначити, видалення яких терм-множин найменше погіршує значення похибки апроксимації.

2. Продемонструвати та пояснити роботу програми на прикладах.

2. Контрольні запитання

1. Можливості спрощення бази правил
2. Міри схожості нечітких множин
3. Метод локальних моделей.

Задачі для самостійного розв'язання: комп'ютерний практикум № 4

Аудиторна робота

Приклад 6.1. Скорочення Бази правил на основі об'єднання термів входів.

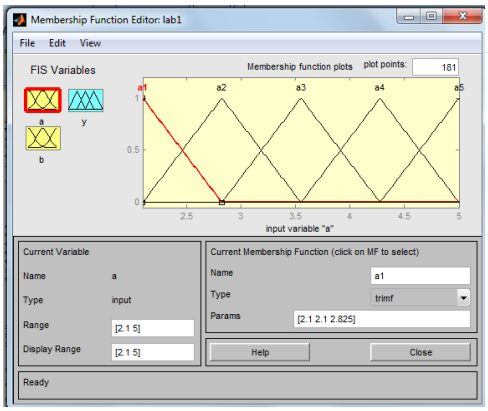
Частина 1: комп'ютерний практикум № 4

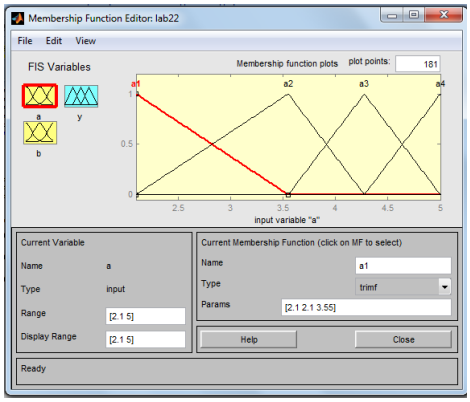
```
clc; close all; clear all
f_name='lab6'; a_t=5;a_beg=2.1;a_end=5;
b_t=4;b_beg=0.5;b_end=3;
mf='trimf'; aprox_func=inline('a-3*b');
format short
n=a_t*b_t; delta_a = (a_end-a_beg)/(a_t-1);
A=a_beg:delta_a:a_end;
delta_b = (b_end-b_beg)/(b_t-1);
B=b_beg:delta_b:b_end;
% [a,b]=meshgrid(A,B);y=aprox_func(a,b)
for i1=1:length(A)
  for i2=1:length(B)
    y(i2,i1)=A(i1)-3*B(i2);
end;  end;
ymin=min(min(y));ymax=max(max(y))
YY=[];
for i=1:b_t
  YY=[YY y(i,:)];
end
[Y, xi]=sort(YY);
ind=[mod(xi-1,a_t)+1; fix((xi-1)/a_t)+1]';
% створення нової СНВ Мамдані
```

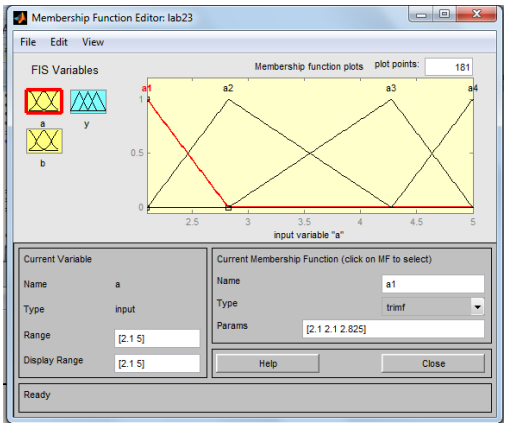
```
ai=newfis('lab6','mamdani','prod','max','prod','max','centroid');
% визначення термів вхідних змінних a та b
ai=addvar(ai,'input','a',[a_beg a_end]);
ai=addvar(ai,'input','b',[b_beg b_end]);
delta_y1=abs(Y(2)-Y(1));delta_y2=abs(Y(n)-Y(n-1));
ai=addvar(ai,'output','y',[ymin-delta_y1 ymax+delta_y2]);
for i=1:a_t
  if strcmp(mf, 'trimf')
    if i==1
     a=a_beg; else a=A(i-1); end
    b=A(i);
    if i==a_t
     c=a_end; else c=A(i+1); end
    MF_A(i,1)=a;MF_A(i,2)=b;MF_A(i,3)=c;
    ai=addmf(ai,'input',1,['a',int2str(i)],mf, [a b c]);
  end
end
for i=1:b_t
  if strcmp(mf,'trimf')
    if i==1
     a=b_beg; else a=B(i-1); end;
    b=B(i);
    if i==b_t
     c=b_end; else c=B(i+1); end;
    MF_B(i,1)=a;MF_B(i,2)=b;MF_B(i,3)=c;
    ai=addmf(ai,'input',2,['b',int2str(i)],mf, [a b c]);
  end
end
% визначення термів вихідної змінної
for i=1:n
  if strcmp(mf,'trimf')
    if i==1
     a=ymin-delta_y1; else a=Y(i-1); end
    b=Y(i);
    if i==n
     c=ymax+delta_y2; else c=Y(i+1); end;
    MF_Y(i,1)=a;MF_Y(i,2)=b;MF_Y(i,3)=c;
    ai=addmf(ai,'output',1,['y',int2str(i)],mf, [a b c]);
  end
end
rullist=ones(n,5);
for i=1:n
    rullist(i,1)=ind(i,1); rullist(i,2)=ind(i,2);
    rullist(i,3)=i;
end
ai=addrule(ai,rullist); writefis(ai,'f_name');
ai=readfis('f_name'); ai=setfis(ai,'defuzzmethod','centroid');
% вхідні параметри для тестування СНВ
AA=2.2:0.5:4.9; BB=0.6:0.5:2.9;
for i =1:length(AA)
   for j=1:length(BB)
y_test(i,j)=(AA(i)-3*BB(j));
```

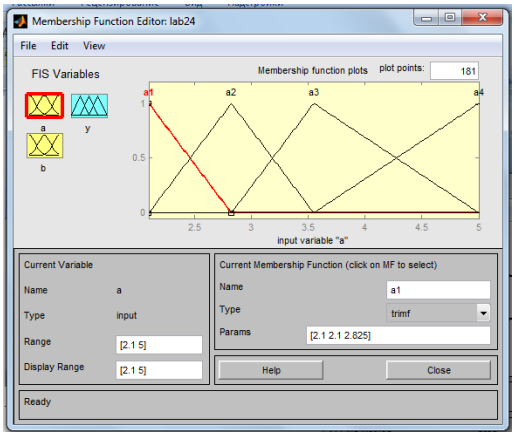
```
end; end
% обчислення нев'язки моделювання функції СНВ при використанні
методу дефазифікації centroid
method_Array=['centroid']; method=cellstr(method_Array);
eps=0;
for i =1:length(AA)
   for j=1:length(BB)
out(i,j)=evalfis([AA(i) BB(j)], ai);
% out(i,j),y_test(i,j)
 eps=eps+abs(out(i,j)-y_test(i,j));
end; end;
eps1=eps/(length(AA)*length(BB))
```

Частина 2: видалення термів







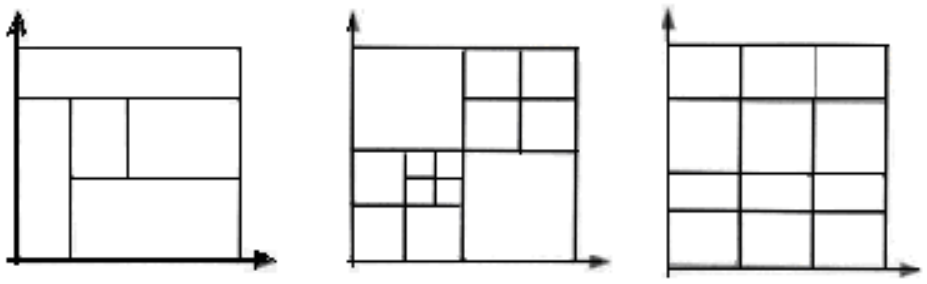


Теоретичні відомості

1. Скорочення бази правил [7, с. 254-270]. Скорочення бази правил є важливим кроком при формуванні нечіткої моделі. Оскільки, із збільшенням кількості входів зростає кількість правил, якими оперувати значно важче, то використання скорочення бази правил не тільки пришвидшує швидкість роботи моделі, а й полегшує керування нею. Скорочення бази правил виконують

декількома методами: 1) шляхом видалення недостатньо важливих правил, нестача яких призводить до незначної похибки, 2) шляхом використання покриття не сіткою, 3) методом локальних моделей. Основна складність процесу налаштування багатовимірних нечітких моделей, які самонавчаються (наприклад, нейронечітких мереж, заснованих на регулярному гіперпрямокутному розбитті простору вхідних значень), полягає у великій кількості параметрів, які підлягають налаштуванню. При цьому ця кількість параметрів стрімко зростає зі збільшенням кількості входів і нечітких множин, які використовують для оцінки їх значень. Дослідженню даної проблеми, названої «прокляттям розмірності», присвячено множину наукових робіт. Один зі способів, запропонованих для її вирішення, полягає у переході від регулярного розбиття вхідного простору до нерегулярного, що складається з прямокутних сегментів. Інший спосіб полягає в тому, щоб відмовитися від розбиття вхідного простору на основі сітки і використовувати розбиття без сітки, наприклад: прямокутне розбиття (k-d tree partition), квадратичне розбиття (quad tree partition). Сутність кожного з цих видів розбиття пояснюється на прикладах, зобажених на рис. 6.1.

Метою застосування розбиття без сітки є зменшення кількості нечітких сегментів. Розбиття вхідного простору буде щільніше в областях, де для модельованої системи поверхня відображення X→Y змінюється більш різко (круті спуски, нерівномірності), і менш щільним в областях з найбільш гладкою поверхнею.



а *б* *в*

Рис. 6.1. Приклади розбиття вхідного простору: прямокутне (а), квадратичне (б), на основі сітки (в)

У межах кожного сегмента розбиття для визначення поверхні використовують тільки одне правило, тому тут доцільно використовувати моделі Такагі–Сугено, в яких висновком кожного правила є функція (зазвичай, лінійна). Прикладом такого правила може служити вираз: if (x_1=A11) and (x_2=A12) then (y=$a_{11}x_1$+$a_{21}x_2$+a_{01}).

Приклад 6.3. Нехай маємо прямокутне розбиття без сітки вхідного простору X=X1×X2, і задані функції належності (рис. 6.2).

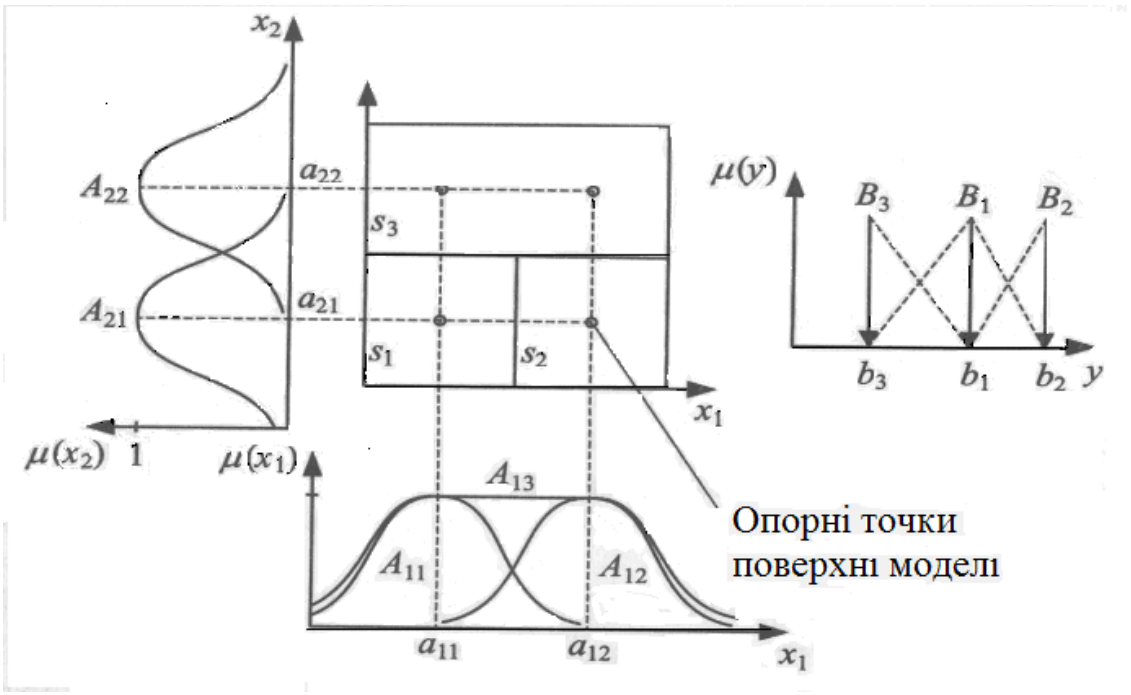


Рис. 6.2. Розбиття вхідного простору без сітки на три сегменти S1 – S3 і завдання ФН нечітких множин

Кожному сегменту може відповідати одне правило, що задає ділянку поверхні моделі, пов'язану з даним сектором. Таким чином, замість чотирьох правил модель (поверхня якої зображена на рис. 6.2) містить три правила вигляду:

R1: ЯКЩО (x1=A11) І (x2=A21) ТО (y=B1)
R2: ЯКЩО (x1=A12) І (x2=A21) ТО (y=B2)
R3: ЯКЩО (x1=A13) І (x2=A22) ТО (y=B3) (6.1)

Можливість застосування великого сегмента S3 вхідного простору X (рис. 6.2) за допомогою тільки одного правила обумовлена тим, що в ньому

використана ФН A13, ядро якої за своєю протяжністю охоплює приблизно всю довжину сегмента.

Для використання розбиття вхідного простору без сітки необхідно попередньо встановити характер зміни поверхні модельованої системи в різних її областях – тільки в цьому випадку можна прийняти обгрунтоване рішення про використання більшої чи меншої щільності розбиття. Необхідну для цього інформацію можна отримати, наприклад, на основі кластерного аналізу вибірки вимірювань «вхід-вихід».

База правил (6.1) є лінгвістично неповною, оскільки в ній присутні не всі можливі комбінації вхідних нечітких множин A_{1i}, A_{2j}. Разом з тим, вона є чисельно повною, внаслідок використання гауссових ФН з необмеженими носіями. При будь-якому вході $(x_1^*, x_2^*) \in X1 \times X2$ буде активовано хоча б одне правило, завдяки чому, незалежно від параметрів ФН (величин лівого та правого розкидів), можна обчислити вихідне значення моделі.

Можливість формування великого сектора S3 обумовлена використанням нечіткої множини A_{13} з відповідним розміром ядра (рис. 6.2). Використання подібних ФН є одним з методів, що дозволяють зменшити кількість правил.

Існує також метод, заснований на зменшенні кількості використаних в моделі нечітких множин, який дозволяє зменшити кількість правил та / або спростити їх форму (*зменшення кількості підумов правил*). Пояснимо сутність даного методу на прикладах.

У разі трикутних множин, якщо нечіткі множини моделі мають близькі модальні значення відносно певних змінних, об’єднання відповідних множин в одну множину не повинно призвести до надмірного зменшення точності моделі, яку можна оцінювати, наприклад, за допомогою суми величин абсолютних похибок за формулою: $I = \sum_{i=1}^{n} \left| y(x_i) - y_m(x_i) \right|$, де n – обсяг вибірки вимірювань «вхід-вихід» параметрів системи. Об'єднання, наприклад, множин $M^* = M \cup L$, можна виконати за формулами $x_M^* = 0.5(x_M + x_L)$, $y_M^* = 0.5(y_M + y_L)$)

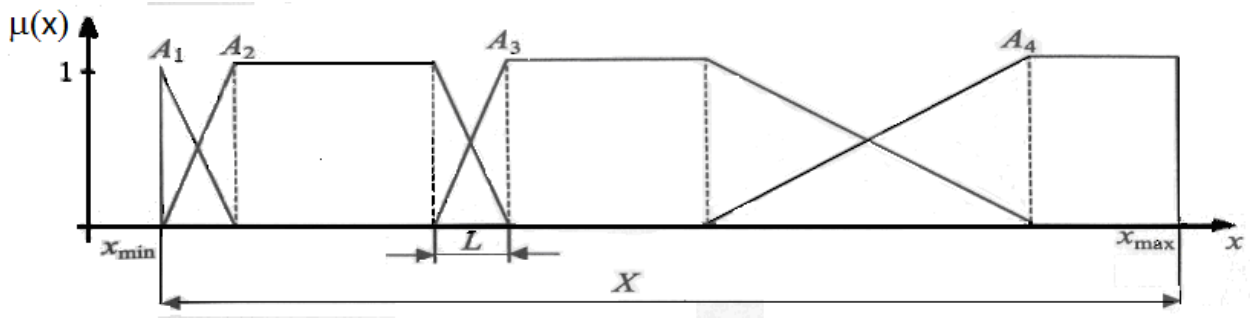
або $\mu_{M*}(x) = \text{sum}\left(\mu_M(x), \mu_L(x)\right)$, $\mu_{M*}(y) = \text{sum}\left(\mu_M(y), \mu_L(y)\right)$. При цьому зменшення кількості множин шляхом їх об'єднання не приведе до істотної зміни точності моделі, хоча кількість правил зменшиться.

У разі трапецієподібних множин об'єднання можливо, якщо вони є суміжними, а їх ядра розташовані близько одне до одного. Поняття «близькості» множин пов'язано не тільки з відстанню L між їх ядрами (рис. 6.3), воно також має враховувати довжину носія множини ($x_{max} - x_{min}$). Таким чином, «близькими» можуть вважатися тільки такі суміжні множини A_i, A_{i+1}, для яких показник *B* відносної близькості (схожості), що виражається формулою (6.2), не перевищує деякого граничного значення:

$$B\left(A_i, A_{i+1}\right) = \frac{L_{A_i, A_{i+1}}}{x_{\max} - x_{\min}} \le \lambda \qquad (6.2)$$

Значення λ вибирає проектувальник моделі: при великих значеннях λ слід очікувати більш помітного зниження точності спрощеної моделі.

Якщо ФН моделі не задовольняють умову розбиття одиниці, то під час налаштування можливі будь-які зміни їх розкиду, у довжині ядра і модальних значень. У цьому випадку в результаті налаштування можна отримати ФН, які у певній мірі перекривають одна одну. Щоб обрати відповідні множини, можна скористатися поняттям міри схожості множин S. Було запропоновано різні міри, кожна з яких відповідає певному критерію подібності.



$$B(A_2, A_3) = \frac{L}{x_{\max} - x_{\min}} \le \lambda$$

Рис. 6.3. Приклад суміжних трапецієподібних нечітких множин, розташованих близько

Однією з них є $S(A_1, A_2)$ – міра подібності двох множин, яка у випадку дискретних ФН задається формулою:

$$S(A_1, A_2) = \frac{|A_1 \cap A_2|}{|A_1 \cup A_2|} = \frac{\sum_{j=1}^{m} \min\left(\mu_{A_1}(x_j), \mu_{A_2}(x_j)\right)}{\sum_{j=1}^{m} \max\left(\mu_{A_1}(x_j), \mu_{A_2}(x_j)\right)},$$

де x_{min}, x_{max} – границі області визначення X (рис. 3); | • | – потужність нечіткої множини; $x_j \in X_D$, X_D – дискретна область значень змінної x.

Приклад 6.4. Розглянемо модель системи з двома входами, одним виходом і лінгвістично повною базою правил:

R1: ЯКЩО (x_1=A1) І (x_2= B1) ТО (y=C1)
R2: ЯКЩО (x_1=A1) І (x_2= B2) ТО (y=C2)
R3: ЯКЩО (x_1=A1) І (x_2= B3) ТО (y=C3)
R4: ЯКЩО (x_1=A2) І (x_2= B1) ТО (y=C4)
R5: ЯКЩО (x_1=A2) І (x_2= B2) ТО (y=C5)
R6: ЯКЩО (x_1=A2) І (x_2= B3) ТО (y=C6)
R7: ЯКЩО (x_1=A3) І (x_2= B1) ТО (y=C7)
R8: ЯКЩО (x_1=A3) І (x_2= B2) ТО (y=C8)
R9: ЯКЩО (x_1=A3) І (x_2= B3) ТО (y=C9), (6.3)

де A_1, A_2, A_3 – нечіткі множини значень входу x_1, B_1, B_2, B_3 – нечіткі множини значень входу x_2, C_1, … , C_9 – нечіткі множини значень виходу y.

Нехай множини $A_2(x_1)$ і $A_3(x_1)$ є схожими і можуть бути об'єднані з отриманням множини $A^*_2(x_1)$, яка потім може бути використана для заміни в правилах множин $A_2(x_1)$ і $A_3(x_1)$ ($A_2 \cong A_2^*, A_3 \cong A_2^*$). В результаті пари правил (R4, R7), (R5, R8), (R6, R9) матимуть однакові умовні частини, і можуть бути об'єднані з використанням операції АБО, що приведе до бази правил вигляду:

R1: ЯКЩО (x_1=A_1) І (x_2=B_1), ТО (y=C_1)
R2: ЯКЩО (x_1=A_1) І (x_2=B_2), ТО (y=C_2)
R3: ЯКЩО (x_1=A_1) І (x_2=B_3), ТО (y=C_3)
R4: ЯКЩО (x_1= A^*_2) І (x_2=B_1), ТО (y=$C_4 \cup C_7$)
R5: ЯКЩО (x_1= A^*_2) І (x_2=B_2), ТО (y=$C_5 \cup C_8$)
R6: ЯКЩО (x_1= A^*_2) І (x_2=B_3), ТО (y=$C_6 \cup C_9$). (6.4)

У розглянутому випадку зменшення кількості нечітких множин на «1» дозволяє зменшити кількість правил на «3». Спрощену модель (6.4) можна вважати допустимою тільки, якщо, в порівнянні з моделлю (6.3) не відбулося значного зменшення її точності.

Тема 3. Нечітка модель Сугено

7. Класифікатори на основі систем нечіткого виведення: нечітка модель Сугено

Мета роботи:

I) ознайомитися 1) з принципами функціонування та навчання *нечіткої моделі Сугено*; 2) з особливостями розв'язання задачі класифікації на базі нечітких моделей, навчитися в середовищі MatLab: розробляти класифікатор в середовищі пакету Fuzzy Logic Toolbox системи MatLab; моделювати нелінійні функції на основі нечіткої моделі Сугено;

II) навчитися розробляти систему нечіткого виведення, здатну розв'язувати задачу класифікації.

Об'єкт дослідження: нечітка модель Сугено на основі пакету Fuzzy Logic Toolbox; класифікатори на основі СНВ.

Питання для опрацювання

1. Розв'язання задачі класифікації за допомогою СНВ: розробка класифікаторів у вигляді СНВ на основі алгоритму Мамдані (при наявності числових та якісних вхідних даних) [3].

Завдання до практичної роботи

1. Вивчити теоретичний матеріал.

2. Послідовно виконати такі завдання до практичної роботи:

2.1. Спроектувати та розробити класифікатор (апроксиматор) у вигляді *нечіткої моделі Сугено*.

Задача класифікації. Побудова нечіткої моделі розв'язання задачі діагностики ґрунтується на формальному поданні характеристик досліджуваної системи (оргагнізму людини) в термінах вхідних (параметри стану хворого) та вихідних (діагнози) лінгвістичних змінних бази правил у СНВ.

Задача полягає в розробці автоматизованої системи обробки інформації для проведення диференційної діагностики. Перелік експериментальних даних, необхідних для виконання роботи, визначається індивідуальним завданням.

2.2. Перевірити модель на здатність розв'язувати задану задачу на прикладах.

Контрольні запитання та завдання

1. Опишіть алгоритм нечіткого виведення Сугено.

2. Дайте коротку характеристику СНВ для розв'язання задачі класифікації.

Задачі для самостійного розв'язання: *проектування та розробка класифікатора*

(база даних Діабет: Додаток В)

№ варіанту	Номери прикладів	№ варіанту	Номери прикладів
1	1÷40	13	240÷280
2	20÷60	14	260÷300
3	40÷80	15	280÷320
4	60÷100	16	300÷340
5	80÷120	17	320÷360
6	100÷140	18	340÷380
7	120÷160	19	360÷400
8	140÷180	20	380÷420
9	160÷200	21	400÷440
10	180÷220	22	420÷460
11	200÷240	23	440÷480
12	220÷260	24	460÷500

Аудиторна робота

Розробити класифікатор для розпізнавання образів (діагнозів) у вигляді СНВ типу Сугено. Кожний образ характеризується властивостями, які приймають числові або якісні значення. Програма генерує відповідну базу знань. При цьому кількість термів вихідної змінної дорівнює кількості класів.

Приклад 7.1. Приклад формування бази правил класифікації рівня захворювання на анемію. *Анемія* – це захворювання, пов'язане із зменшенням кількості червоних кров'яних тілець (еритроцитів).

Виділяють такі *ступені тяжкості захворювання на анемію*

1) легка:

- рівень гемоглобіну нижче норми, але вище 90 г/л;
- кількість еритроцитів в крові становить:

у чоловіка становить $3 \div 4{,}5 \cdot 10^{12}$; у жінки: $3 \div 3{,}7 \cdot 10^{12}$;

2) середня:

- рівень гемоглобіну 90–70 г/л;
- кількість еритроцитів в крові становить $2 \div 3 \cdot 10^{12}$;

3) важка:

- рівень гемоглобіну 30–70 г/л;
- кількість еритроцитів в крові становить $1 \div 2 \cdot 10^{12}$;

4) норма:

- рівень гемоглобіну

для чоловіка: 135–160 г/л ($4{,}5 \div 5{,}5 \cdot 10^{12}$); для жінки: 120–140 г/л;

- кількість еритроцитів в крові становить $3{,}7 \div 4{,}7 \cdot 10^{12}$.

База правил:

```
R1: Якщо (x1=тяжка) і (x2=тяжка) То (y=тяжка)
R2: Якщо (x1=тяжка) і (x2=середня) То (y=тяжка)
R3: Якщо (x1=тяжка) і (x2=легка) То (y=тяжка)
R4: Якщо (x1=тяжка) і (x2=норма) То (y=тяжка)
R5: Якщо (x1=середня) і (x2=тяжка) То (y=тяжка)
R6: Якщо (x1= середня) і (x2= середня) То (y= середня)
```

```
R7: Якщо (x1= середня) і (x2=легка) То (y= середня)
R8: Якщо (x1= середня) і (x2=норма) То (y= середня)
R9: Якщо (x1= легка) і (x2=тяжка) То (y= тяжка)
R10: Якщо (x1= легка) і (x2=середня) То (y= середня)
R11: Якщо (x1= легка) і (x2=легка) То (y= легка)
R12: Якщо (x1= легка) і (x2=норма) То (y= легка)
R13: Якщо (x1= норма) і (x2=тяжка) То (y= тяжка)
R14: Якщо (x1= норма) і (x2=середня) То (y= середня)
R15: Якщо (x1= норма) і (x2=легка) То (y= легка)
R16: Якщо (x1= норма) і (x2=норма) То (y= норма)
```

https://habrahabr.ru/post/113020/

Приклад 7.2. Приклад моделі Сугено розв'язання задачі класифікації

```
clc; clear all; close all; f_name='lab_7';
numOfVars = 4; numOfDiag = 4;
dataMin = [ 0 0 8.8 0;
            12 0.1 5.5 0;
            20 0.3 2.5 60;
            30 0.3 0 190];
dataMax = [12 0.1 20 40;
            20 0.2 8.8 60;
            30 1 5.5 190;
            50 1 2.5 300];
for i = 1:numOfVars
    varMin(i) = min(dataMin(:, i));
    varMax(i) = max(dataMax(:, i));
end;
nameOfVars = ['volume'; 'depth '; 'T3    '; 'ATTG  '];
dataMid = (dataMax + dataMin)/2;
delta_h = (dataMid - dataMin)/(100);
dataMin = dataMin-delta_h; dataMax = dataMax+delta_h;
fs=newfis(f_name,'sugeno', 'prod', 'probor', 'prod', 'sum',
'wtaver');
for inVar = 1:numOfVars
    fs.input(inVar).name = nameOfVars(inVar, :);
    fs.input(inVar).range=[varMin(inVar) varMax(inVar)];
    for dat = 1:numOfDiag
        fs.input(inVar).mf(dat).name=[nameOfVars(inVar, :),
int2str(dat)];
        fs.input(inVar).mf(dat).type='trimf';
        fs.input(inVar).mf(dat).params=[dataMin(dat, inVar)
dataMid(dat, inVar) dataMax(dat, inVar)];
    end; end
 nameOfDiag = ['giperplaz  '; 'norm       '; 'difuz-zob  ';
'giperterioz';];
fs.output(1).name = 'Diagnosis'; fs.output(1).range=[0 numOfDiag];
for diag = 1:numOfDiag
    fs.output(1).mf(diag).name = nameOfDiag(diag, :);
    fs.output(1).mf(diag).type='constant';
    fs.output(1).mf(diag).params= diag;
end
```

```
   rullist = ones(numOfDiag, numOfVars + 3);
   for i=1:numOfDiag
       for j=1:numOfVars
           rullist(i,j) = i;
       end;
       rullist(i, numOfVars + 1) = i;
   end
   fs=addrule(fs,rullist);  writefis(fs, f_name);

ai = readfis(f_name);
disp('+-------------------------+');
disp('|Діагностування захворювань|');
disp('+-------------------------+');
volume = input('Введіть симптом V = ');
depth = input('Введіть симптом d = ');
t3 = input('Введіть симптом T3 = ');
attg = input('Введіть симптом ATTG = ');
ai = setfis(ai,'defuzzmethod', 'wtaver');
outt = evalfis([volume depth t3 attg], ai);
disp(sprintf('Diagnoz is %s', nameOfDiag(outt, :)));
```

Приклад 7.3. Приклад моделі Сугено розв’язання задачі класифікації

```
script
Data=[361 320 401 350 340 229 291 229 260 264 170 152 167 267 151
161 181 174;
970 648.4 1115 604.5 625 1040 840 1040 796 762 566 290 604.5 554
110 141 150 123;
58 60 60 60.6 49 58 57 58 61 59 61 60 60.5 40.1 35.2 45.3 36.5
39.2;
4.5 3.4 5.3 3.5 3.55 3.6 4 3.6 2.4 4.2 4.1 4.5 3.4 1.4 1.2 2.2
1.72 2.31;
17.6 13.1 20.5 14 11.4 14 11 14 3.65 4.5 7.9 12.7 13.35 15.1 8.2
8.7 9.2 6.4;
40 25.6 37.2 26 34 33 31 33 34 24 27 15 25.7 20.3 11 13.5 15.1
14.7];
% точки БД
Dind = [1 2 3 3; 4 5 5 5; 6 7 8 8; 9 10 11 11; 12 13 14 14; 15 16
17 18]
n=6;% atributi diagnosa
m=6;% koldiagn
max_k=4;% максим. кількість точок, що описують діагноз
f_name='l_9_1Sug';
perem=n;koldiag=m;
AllMin=zeros(perem); AllMax=zeros(perem);

for ind = 1:1:perem
AllMin(ind)=min(Data(ind,1:1:18)) %18-загальна кількісcть точок в
БД
AllMax(ind)=max(Data(ind,1:1:18))
end
DataMid = zeros(perem, koldiag);
```

```
DataMax=zeros(perem, koldiag); DataMin=zeros(perem, koldiag);

for pokaz=1:1:perem;
 for din=1:1:koldiag
     DataMin(pokaz,din)=Data(pokaz,Dind(din,1));
     DataMax(pokaz,din)=Data(pokaz,Dind(din,1));

for i=Dind(din, 1:1:max_k)
 if (DataMin(pokaz,din)>Data(pokaz,i))
     DataMin(pokaz,din)= Data(pokaz,i);
 end
 if (DataMax(pokaz,din)<Data(pokaz,i))
     DataMax(pokaz,din)= Data(pokaz,i);
end;end; end; end

MfN=['a' 'b' 'c' 'e' 'f' 'g' 'h']
DN=['1' '2' '3' '4' '5' '6' ]
DataMid=(DataMax+DataMin)/2;
delta_h=(DataMid-DataMin)/100000;
DataMin=DataMin-delta_h;
DataMax=DataMax+delta_h;
fs=newfis(f_name,'sugeno', 'min', 'max', 'prod', 'sum', 'wtaver');
for inp=1:1:perem
 fs.input(inp).name=MfN(inp);
 fs.input(inp).range=[AllMin(inp) AllMax(inp)];
 for dat=1:1:koldiag
  fs.input(inp).mf(dat).name=[MfN(inp),DN(dat)];
  fs.input(inp).mf(dat).type='trimf';
  fs.input(inp).mf(dat).params=[DataMin(inp,dat)   DataMid(inp,dat)
DataMax(inp,dat)];
end; end
fs.output(1).name='Diagnosis';
fs.output(1).range=[0 perem+0.5];
for ind = 1:1:koldiag
 fs.output(1).mf(ind).name=num2str(ind);
 fs.output(1).mf(ind).type='constant';
 fs.output(1).mf(ind).params=ind;
end
rullist=ones(m, n+1+2);
  for i=1:m
  for j=1:n
    rullist(i,j)=i;
  end;
  rullist(i,n+1)=i;
  end
  fs=addrule(fs,rullist);  writefis(fs,'l_9_1Sug');
  sug_FuSy=readfis(f_name); % showfis (sug_FuSy);
neviazka=0;kpr=0;kolichestvo=0;
for k=1:1:koldiag
for i=Dind(k)
 if kpr==i
 else
```

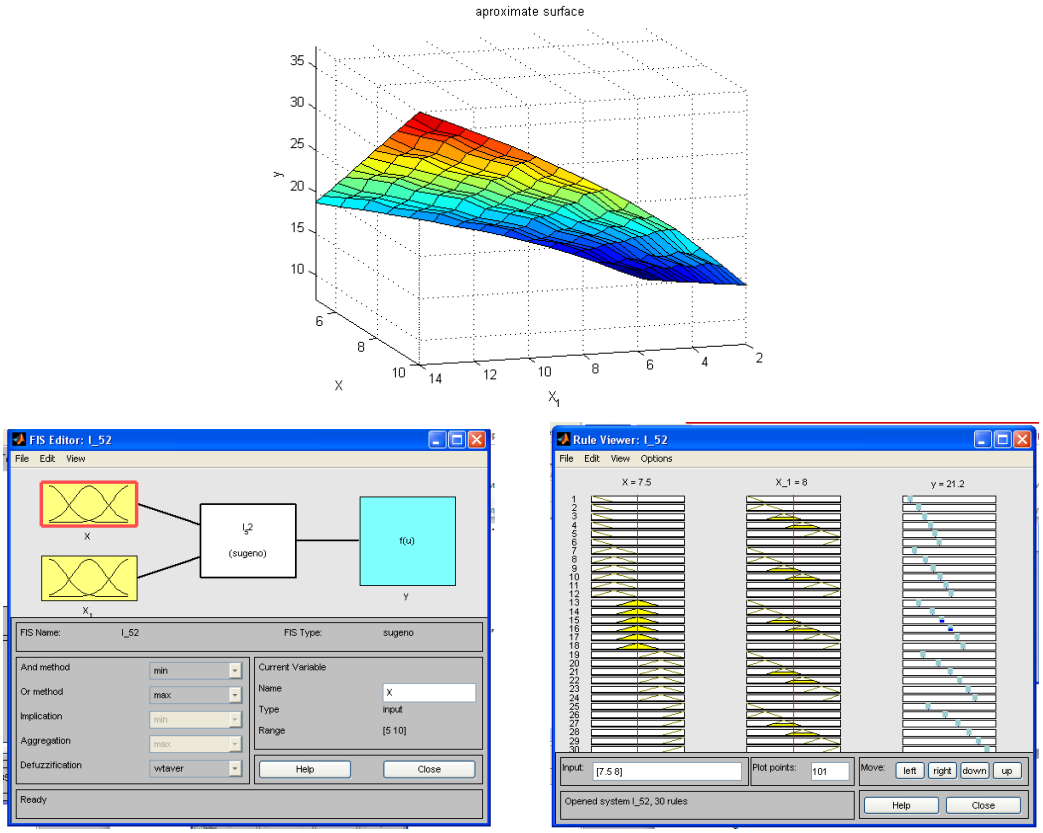
```
 Data(1:1:perem, i)
neviazka=neviazka+abs(round(evalfis(Data(1:1:perem,i),sug_FuSy))  -
k);
 kolichestvo=kolichestvo+1;
 end
 kpr=i;
end; end
neviazka=neviazka/kolichestvo
fuzzy 'l_9_1Sug'
```

Приклад 7.4. Приклад моделі Сугено розв'язання задачі апроксимації

```
script  %function [X,X_1,Y,out, ep]=lab_9_2
clc;clear all;close all;
%початкові дані
x_t=5; x_beg=5; x_end=10; x_t_1=6; x_beg_1=2; x_end_1=14;
f_name='l_9_2';
n=x_t; %кількість термів X
m=x_t_1;%кількість термів X_1
nm=n*m;
%формування вхідних чітких векторів X і X_1
X=x_beg:(x_end-x_beg)/(n -1):x_end;
delta_x=(x_end-x_beg)/(n-1)
X_1=x_beg_1:(x_end_1-x_beg_1)/(m-1):x_end_1;
delta_x1=(x_end_1-x_beg_1)/(m-1)
% Формування масиву Y (вихідних даних)
z=0;
for i=1:n
 for j=1:m
  z=z+1; Y(z)=X(i)*sqrt(X_1(j));
end; end
k=1;data=zeros(m*n,3)
for i=1:n
 for j=1:m
   data(k,1)=X(i);data(k,2)=X_1(j); data(k,3)=Y(k); k=k+1;
end;end; % data
Y2=sort(Y);
%визначення термів вихідної змінної 'Y'
ymin=min(Y);ymax=max(Y);
delta_y=(ymax-ymin)/nm;
% Створення нової системи нечіткого виведення типу Мамдані
ai=newfis(f_name,'sugeno', 'min', 'max', 'prod', 'sum', 'wtaver');
%визначення термів вхідних змінних X та X_1
ai= addvar (ai,'input','X', [x_beg x_end]);
for i=1:n
if i==n
 c=x_end+delta_x; else c=X(i+1)
end;
if i==1
 a=x_beg -delta_x; else a=X(i-1)
end; b=X(i);
ai=addmf (ai,'input',1, int2str(i),'trimf', [a b c])
```

```
end;
ai = addvar (ai,'input','X_1', [x_beg_1 x_end_1])
for i=1:m
 if i==m
 c=x_end_1 + delta_x1; else c=X_1(i+1)
 end;
if i==1
 a=x_beg_1-delta_x1; else a=X_1(i-1)
end; b=X_1(i);
ai=addmf (ai,'input',2, int2str(i),'trimf', [a b c])
end;
%визначення термів вихідної змінної 'Y'
ai= addvar (ai,'output','y', [ymin ymax]);
for i=1:nm
 if i==nm
  c=ymax+delta_y;
 else
    c=Y2(i+1);
 end;
 if i==1
  a=ymin-delta_y;
 else
  a=Y2(i-1)
 end;  b=Y2(i);
ai= addmf (ai,'output',1, ['y', int2str(i)],'constant', b);
end;
% Формування масиву правил на основані спостережень за графіком
rullist=ones(nm,5); z=0;
for i=1:n
 for j=1:m
  z=z+1; rullist(z,1)=i; rullist(z,2)=j;
 for k=1:nm
  if (Y((i-1)*m+j)==Y2(k)) %disp('U1 = ');
 break;
 end; end
 rullist(z,3)=k; rullist(z,4)=1;
end; end;
ai= addrule(ai, rullist);
figure (2), gensurf (ai);title ('aproximate surface')
writefis (ai,f_name)
% параметри для нечіткого виведення
X_check=6:0.2:9; rr=length(X_check); X_1_check=9;
Y_check(1:rr)=X_check(1:rr)*sqrt(X_1_check)
%обчислення нев'язки моделювання функції СНВ
  ai=readfis(f_name);  ep=0; % showfis(ai);
for i = 1:rr
 out(i)=evalfis([X_check(i) X_1_check], ai);
 ep=ep+abs(out(i)-Y_check(i));
end
ep=ep/rr; fuzzy 'l_9_2'
```

Результат роботи:



Теоретичні відомості

1. Розв’язання задачі класифікації за допомогою СНВ [3]

Класифікацією називається процес визначення відповідності об’єкта D одному з класів d_1, ..., d_s із врахуванням його властивостей (параметрів стану) x_1, ..., x_n (рис. 7.1). До задач класифікації відноситься і задача медичної діагностики.

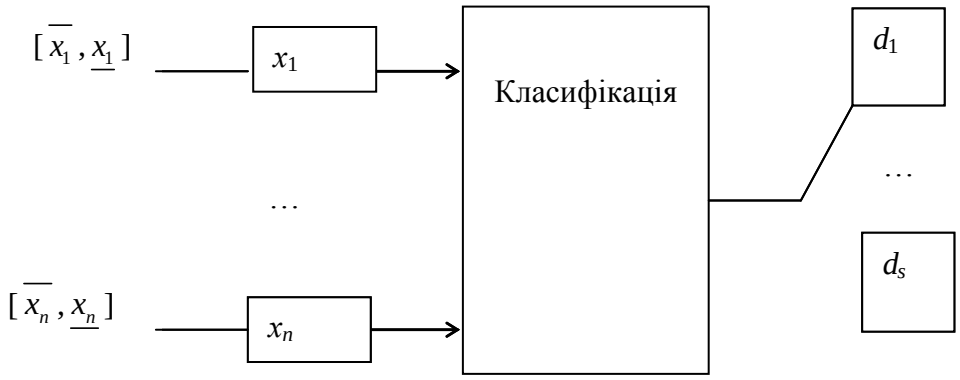


Рис. 7.1. Схема розв’язання задачі класифікації за допомогою СНВ

Автоматичним класифікатором називають програмно-апаратний комплекс, здатний самостійно розв'язати конкретну задачу класифікації. Деякі класифікатори розробляють на основі системи нечіткої логіки. В цьому випадку в якості алгоритму нечіткого виведення можуть бути використані алгоритми Мамдані та Сугено.

Розробка класифікаторів у вигляді СНВ Мамдані

1.1. При наявності числових вхідних даних. В процесі вирішення задачі класифікації (на прикладі задачі діагностики) розглянемо задачу з такими початковими даними:

1) множина розв'язків (діагнозів): D={ d_j, $j=1,...,m$ };

2) множина вхідних параметрів (симптомів): X={ $x_i, i=1,...,n$ };

3) діапазони кількісної зміни кожного вхідного параметру: $x_i \in [\underline{x_i}, \overline{x_i}]$, $i=1,...,n$;

4) функції належності, які дозволяють зображувати вхідні параметри та вихідний параметр у вигляді нечітких множин: $a_i^p = \int_{\underline{x_i}}^{\overline{x_i}} \mu^{a_i^p}(x_i)/x_i$; $d_j = \int_{\underline{d}}^{\overline{d}} \mu^{d_j}(d)/d$,

де $\mu^{a_i^p}(x_i)$ – ФН вхідного параметра $x_i \in [\underline{x_i}, \overline{x_i}]$, для терму $a_i^p \in A_i$, $p=1,...,l_i$; $i=1,...,n$; $\mu^{d_j}(d)$ – ФН вихідного параметра $y \in [\underline{y}, \overline{y}]$ для терму $d_j \in D$, $j=1,...,m$;

5) матриця знань.

Зупинимося більш детально на формуванні матриці знань. Матриця знань – це таблиця даних (табл. 7.1), сформована за наступними правилами:

1. Розмір матриці дорівнює $(n+1)\times N$, де $(n+1)$ – число стовпчиків, $N=k_1+\ldots+k_m$ – число рядків.

2. Перші n стовпчиків матриці відповідають вхідним параметрам $x_i \in [\underline{x_i}, \overline{x_i}]$, $i=1,...,n$, а $(n+1)$-й стовпчик відповідає значенням d_j, $j=1,...,m$ вихідного параметру y.

3. Перші k_1 рядків відповідають значенню вихідного параметра d_1, наступні k_2 рядків – значенню d_2, ..., останні k_m рядків – значенню d_m.

4. Елемент a_i^{jp}, який розташований на перетині i-го стовпчика й jp-го рядка, відповідає лінгвістичній оцінці параметра x_i в рядку нечіткої бази знань за номером jp. При цьому лінгвістична оцінка a_i^{jp} обирається з терм-множини, яка відповідає параметру x_i, тобто $a_i^{jp} \in A_i$; $j=1,...,m$; $i=1,...,n$; $p=1,...,k_j$.

Одержана таким чином матриця знань визначає базу знань СНВ:

ЯКЩО $(x_1 = a_1^{11})$ І...І $(x_n = a_n^{11})$ АБО … АБО $(x_1 = a_1^{1k_1})$ І...І $(x_n = a_n^{1k_1})$ ТО $y=d_1$ …

ЯКЩО $(x_1 = a_1^{m1})$ І...І $(x_n = a_n^{m1})$ АБО ... АБО $(x_1 = a_1^{mk_m})$ І...І $(x_n = a_n^{mk_m})$ ТО $y=d_m$

Тут a_i^{jp} – лінгвістична оцінка вхідного параметра x_i в p-му рядку j-ої диз'юнкції, яка обирається з відповідної терм-множини A_i; d_j ($j=1,...,m$) – лінгвістична оцінка вихідного параметра y, яка визначається з терм-множини D; k_j – кількість правил, які визначають значення вихідного параметра $y=d_j$.

Таблиця 7.1

Матриця знань

Номер вхідної комбінації значень	Вхідні параметри			Вихідний параметр
	x_1	...	x_n	
11 ... $1k_1$	a_1^{11} … $a_1^{1k_1}$	…	a_n^{11} … $a_n^{1k_1}$	d_1
…	…	…	…	…
m_1 … mk_m	a_1^{m1} … $a_1^{mk_m}$	…	a_n^{m1} … $a_n^{mk_m}$	d_m

Отже, на основі визначених даних необхідно розробити класифікатор у вигляді СНВ, яка дозволяє фіксованому вектору вхідних параметрів $\mathbf{x}^*=[x_1^* \ .... \ x_n^*]^T$ поставити у відповідність розв'язок $y^* \in D$.

Приклад 7.5. Наведемо приклад визначення такого класифікатора для розпізнавання образів-діагнозів. Кожний образ характеризується властивостями (симптомами), які приймають числові значення (табл. 7.2). Класифікатор має вигляд СНВ Мамдані в середовищі MatLab на основі Fuzzy Logic Toolbox.

Таблиця 7.2

Експериментальні дані

b	*c*	*e*	*f*	*g*	*h*	*Діагноз*
361	970	58	4,5	40	17,6	D1
320	648,4	60	3,4	25,6	13,1	D1
401	1115	60	5,3	37,2	20,5	D1
350	604,5	60,6	3,5	26	14	D2
340	625	49	3,55	34	11,4	D2
151	110	35,2	1,2	11	8,2	D3
161	141	45,3	2,2	13,5	8,7	D3
181	150	36,5	1,72	15,1	9,2	D3
174	123	39,2	2,31	14,7	6,4	D3

```
script
format long
Data=... % точки БД
[361   320   401    350     340   151     161  181  174 ;
 970  648.4  1115   604.5   625    110   141   150  123 ;
  58   60    60     60.6    49     35.2  45.3  36.5  39.2 ;
 4.5   3.4   5.3    3.5     3.55   1.2     2.2  1.72 2.31 ;
  40   25.6 37.2    26      34     11   13.5   15.1  14.7 ;
 17.6 13.1  20.5    14      11.4   8.2   8.7   9.2   6.4] ;
Dind = [1 2 3 3 ; 4 5 5 5; 6 7 8 9]
n=6; m=3;
max_k=4; % 4-максимальна кількість точок, що описують діагнози
f_name='lab_9_3'; perem=n; koldiag=m;
for ind = 1:1:perem
  AllMin(ind)=min(Data(ind,1:1:9)); %9-загальна кількісссть точок в
БД
  AllMax(ind)=max(Data(ind,1:1:9));
end
 for pokaz=1:1:perem
  for din=1:1:koldiag
    DataMin(pokaz,din)=Data(pokaz,Dind(din,1));
    DataMax(pokaz,din)=Data(pokaz,Dind(din,1));
    for i=Dind(din, 1:1:max_k)
     if    (DataMin(pokaz,din)>Data(pokaz,i))    DataMin(pokaz,din)=
Data(pokaz,i);
     end
     if (DataMax(pokaz,din)<Data(pokaz,i))
        DataMax(pokaz,din)= Data(pokaz,i);
     end
```

```
 end;  end; end;
MfN=['b' 'c' 'e' 'f' 'g' 'h'];
DataMid=(DataMax+DataMin)/2;
delta_h=(DataMid-DataMin)/(1e5);
DataMin=DataMin-delta_h; DataMax=DataMax+delta_h;
fs=newfis(f_name,'mamdani','min','max','min','max','centroid');
for inp=1:1:perem
    fs.input(inp).name=MfN(inp);
    fs.input(inp).range=[AllMin(inp) AllMax(inp)];
    for dat=1:1:koldiag
        fs.input(inp).mf(dat).name=[MfN(inp),int2str(dat)];
        fs.input(inp).mf(dat).type='trimf';
        fs.input(inp).mf(dat).params=[DataMin(inp,dat)
DataMid(inp,dat) DataMax(inp,dat)];
end; end
fs.output(1).name='Diagnosis';
fs.output(1).range=[0 koldiag+1.0];
for ind = 1:1:koldiag
    fs.output(1).mf(ind).name=['diagn_',int2str(ind)];
    fs.output(1).mf(ind).type='trimf';
    fs.output(1).mf(ind).params=[ind-1.0 ind ind+1.0];
end
   rullist=ones(m, n+1+2);
   for i=1:m
     for j=1:n
      rullist(i,j)=i;
     end;
     rullist(i,n+1)=i;
   end
   fs=addrule(fs,rullist);
   writefis(fs, f_name); fuzzy  'lab_9_3'
```

1.2. При наявності якісних вхідних даних [3]

Приклад 7.6. Наведемо приклад класифікатора для задачі діагностики «*Опіки шкіри при ДТП*», схему розв'язання якої зображено на рис. 7.2.

Постановка задачі: визначити за вхідними якісними даними (симптомами) діагноз, який їм відповідає. При діагностиці враховують:

1. *Вхідні лінгвістичні змінні*: *a*=ушкодження_шкіри (наявність ушкодження шкіри або його відсутність), *b*=вплив_на_шкіру_високої_ температури_або_хімічних_речовин (наявність впливу високої температури або впливу хімічних речовин, або їх відсутність), *c*=вплив_ на_шкіру_кислот_або_лугів (наявність впливу кислот або впливу лугів, або їх

відсутність), *d* (наявність почервоніння шкіри, пухирів, омертвілих ділянок або відсутність зазначених ознак).

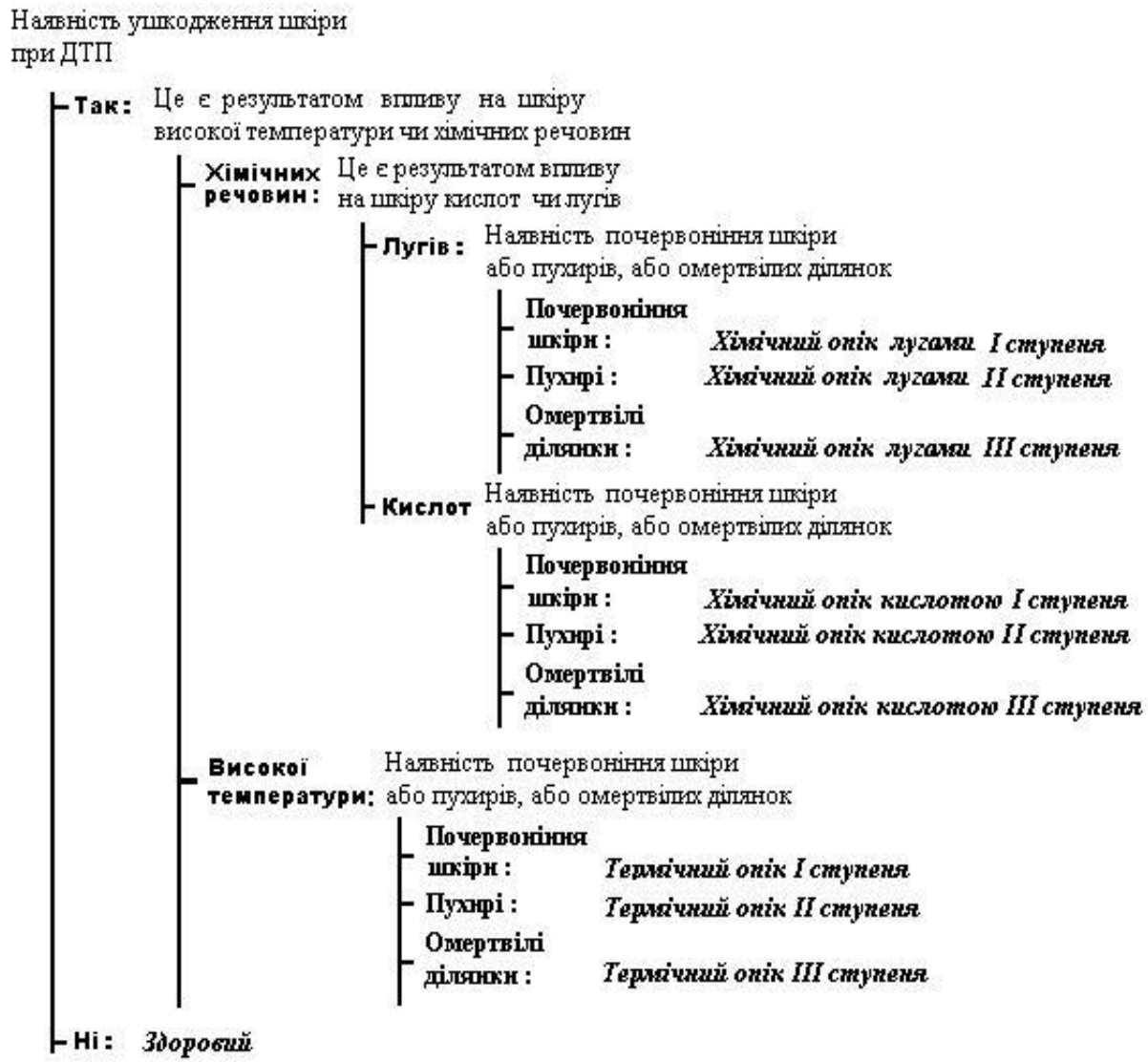
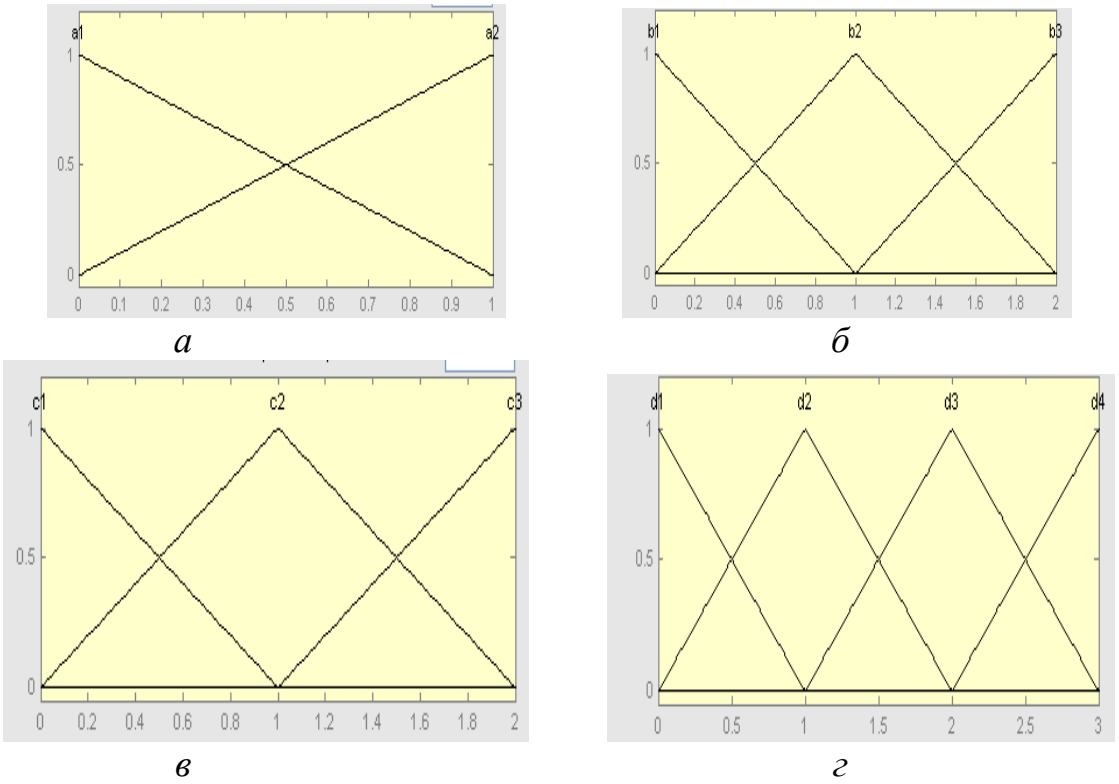


Рис. 7.2. Схема розв'язання ситуаційної задачі «Опіки шкіри при ДТП»

2. *Вихідну лінгвістичну змінну* D, яка приймає одне із множини можливих значень {Здоровий, Хімічний опік лугами I (II, III) ступеня, Хімічний опік кислотою I (II, III) ступеня, Термічний опік I (II, III) ступеня}.

Фазифікація вхідних і вихідних змінних. В якості терм–множин вхідної змінної *a* будемо використовувати множину {*a*1, *a*2} з функціями належності для термів, зображенними на рис. 7.3.*а*; вхідної змінної *b* – множину {*b*1, *b*2, *b*3} з ФН для термів, зображенними на рис. 7.3.*б*; вхідної змінної *c* – множину {*c*1, *c*2, *c*3} з ФН для термів, зображенними на рис. 7.3.*в*; вхідної змінної *d* –

множину {*d*1, *d*2, *d*3, *d*4} з ФН для термів, зображенними на рис. 7.3.*г*. В якості терм–множини вихідної змінної D будемо використовувати множину {D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8, D_9, D_{10}} з симетричними трикутними ФН (рис. 7.4).



а *б*

в *г*

Рис. 7.3. Графіки функцій належності для термів вхідних змінних *a*, *b*, *c*, *d*

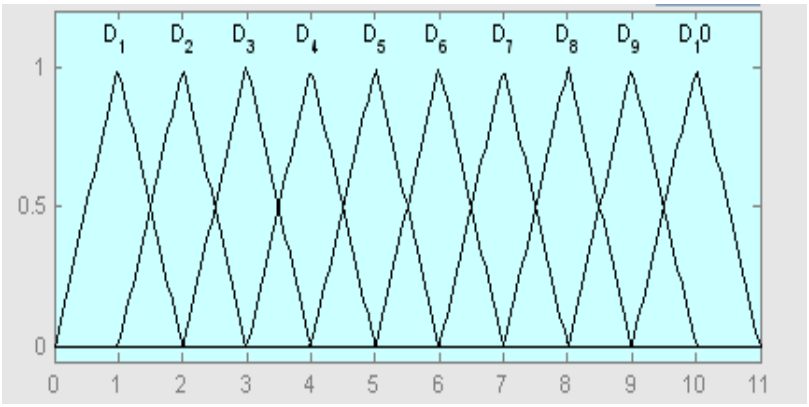


Рис. 7.4. Графіки ФН для термів вихідної змінної D

Формування нечіткої асоціативної матриці, яка описує відношення вхід-вихід (табл. 7.3). Враховуючи значення вершин (центральних точок) симетричних трикутних ФН термів вхідних змінних, зафіксованих в табл. 7.3, зобразимо дані табл. 7.3 у вигляді табл. 7.4.

Таблиця 7.3

Нечітка асоціативна матриця

a	*b*	*c*	*d*	*Діагноз*
a1	b1	c1	d1	D1 – Здоровий
a2	b2	c2	d2	D2 – Хімічний опік лугами I ступеня
a2	b2	c2	d3	D3 – Хімічний опік лугами II ступеня
a2	b2	c2	d4	D4 – Хімічний опік лугами III ступеня
a2	b2	c3	d2	D5 – Хімічний опік кислотою I ступеня
a2	b2	c3	d3	D6 – Хімічний опік кислотою II ступеня
a2	b2	c3	d4	D7 – Хімічний опік кислотою III ступеня
a2	b3	c1	d1	D8 – Термічний опік I ступеня
a2	b3	c1	d2	D9 – Термічний опік II ступеня
a2	b3	c1	d3	D10 – Термічний опік III ступеня

Таблиця 7.4

Нечітка асоціативна матриця

a	*b*	*c*	*d*	*Діагноз*
0	0	0	0	D1
1	1	1	1	D2
1	1	1	2	D3
1	1	1	3	D4
1	1	2	1	D5
1	1	2	2	D6
1	1	2	3	D7
1	2	0	1	D8
1	2	0	2	D9
1	2	0	3	D10

Приклад програми формування класифікатора у вигляді відповідної СНВ Мамдані в середовищі MatLab на базі пакету Fuzzy Logic Toolbox.

```
script
format long
Data= ...  % точки БД
[0 1 1 1 1 1 1 1 1 1;
 0 1 1 1 1 1 1 2 2 2;
 0 1 1 1 2 2 2 0 0 0;
 0 1 2 3 1 2 3 1 2 3 ];
Dind = [1; 2; 3; 4; 5; 6; 7; 8; 9; 10]
n=4;    %кількість атрибутів, які описують діагноз
m=10; %кількість діагнозів, що фігурують в БД, та загальна
кількість точок в ній
f_name='lab_Mamdani'; perem=n; koldiag=m;
for ind = 1:1:perem
AllMin(ind)=min(Data(ind,1:1:m));
AllMax(ind)=max(Data(ind,1:1:m));
```

```
end
for pokaz=1:1:perem
    for din=1:1:koldiag
        DataMin(pokaz,din)=Data(pokaz,Dind(din,1));
        DataMax(pokaz,din)=Data(pokaz,Dind(din,1));
    end
end
DataMin=DataMin-1; DataMax=DataMax+1
MfN=['a' 'b' 'c' 'd']
TERM=[2; 3; 3; 4] % кількість термів, що описують симптом
DataMid=(DataMax+DataMin)/2;
delta_h=(DataMid-DataMin)/1000000;
DataMin=DataMin-delta_h; DataMax=DataMax+delta_h;
fs=newfis(f_name, 'mamdani','min','max','min','max', 'centroid');
for inp=1:1:perem
    fs.input(inp).name=MfN(inp);
    fs.input(inp).range=[AllMin(inp) AllMax(inp)];
    if (TERM(inp)~=2)
      for dat=1:1:TERM(inp)
        fs.input(inp).mf(dat).name=[MfN(inp),int2str(dat)];
        fs.input(inp).mf(dat).type='trimf';
        if ((dat==3)&(TERM(inp)==3))
          fs.input(inp).mf(dat).params=[DataMax(inp,dat)-1
DataMax(inp,dat) DataMax(inp,dat)+1];
        else
          fs.input(inp).mf(dat).params=[DataMin(inp,dat)
DataMid(inp,dat) DataMax(inp,dat)];
      end;  end
    end
    if (TERM(inp)==2)
     for dat=1:1:TERM(inp)
      fs.input(inp).mf(dat).name=[MfN(inp), int2str(dat)];
      fs.input(inp).mf(dat).type='trimf';
      if (dat==1)
        fs.input(inp).mf(dat).params=[DataMin(inp,dat)
DataMin(inp,dat)+1  DataMax(inp,dat)];
      end;
      if (dat==2)
       fs.input(inp).mf(dat).params=[DataMin(inp,dat)
       DataMax(inp,dat)-1 DataMax(inp,dat)];
      end;
     end
  end;
end
fs.output(1).name='Diagnos';
fs.output(1).range=[0 koldiag+1.0];
for ind = 1:1:koldiag
  fs.output(1).mf(ind).name=['D_', int2str(ind)];
  fs.output(1).mf(ind).type='trimf';
  fs.output(1).mf(ind).params=[ind-1.0 ind ind+1.0];
end
rullist=ones(m, n+1+2);
```

```
for i=1:koldiag
   for j=1:n
       rullist(i,j)=Data(j,i)+1;
   end;
     rullist(i,n+1)=i;
end
 fs=addrule(fs,rullist); writefis(fs, f_name);
```

8. Порівняння ефективності використання алгоритмів нечіткого виведення Мамдані та Сугено 0-ого порядку

Мета роботи: ознайомитися з алгоритмами нечіткого виведення Мамдані та Сугено 0-ого порядку. *Об'єкт дослідження* – алгоритми нечіткого виведення Мамдані та Сугено 0-ого порядку.

Питання для опрацювання

1. Алгоритми нечіткого виведення Мамдані та Сугено 0-ого порядку [2].

Завдання

1. Реалізувати алгоритми виведення Мамдані та Сугено 0-ого порядку.
2. Порівняння алгоритмів виконати при таких умовах:
- апроксимація функції проводиться на відрізку [-1, 1];
- функція, яку апроксимують задається набором значень (x^k, $y^k=f(x^k)$), k=1,...,К, при цьому точки x^k розташовані на однаковій відстані;
- ФН мають вигляд трикутних функцій (Приклад 10.1) або функцій Гауса;
- кількість правил n задано заздалегідь.
3. Результат порівняння апроксимації функцій на основі заданих алгоритмів навести у вигляді таблиці

2. Контрольні запитання

1. Опишіть алгоритми виведення Мамдані та Сугено 0-ого порядку

Задачі для самостійного розв'язання: моделювання відношення «вхід-вихід», заданого на основі функції від однієї змінної

№	$f(x)$	[*a, b*]
1	$2x+\sqrt{x}$	1; 3
2	$\sqrt[3]{x}$	3; 4
3	$(1+x)^3$	2; 3
4	$1/\sqrt[3]{x}$	0,1; 1,0
5	$1/\sqrt{x}$	6; 8
6	$(2x+1)/\sqrt{x}$	8; 9
7	ln(1+*x*)/(1-x)	3; 6
8	$\sqrt[5]{x}\cdot\cos(x)$	1; 1.5
9	*x* + sin(*x*)	0; π/2
10	$e^{(1+x)}$	2,5; 3,0
11	e^{-x}	0,5; 1,0
12	$1/(1+e^{(1+x)})$	3; 4
13	(e^{2x})*sqrt(x)	1; 3
14	1+ $\ln^2(1-x)$	0,4; 0,9
15	$\ln(1+x)^2$	1; 3

№	$f(x)$	$[a, b]$
16	$\sqrt[4]{x}$	0,5; 2
17	$(2+x)/\sqrt[3]{x}$	5; 7
18	$\frac{1}{\sqrt{1+x}}$	0; 2
19	$(1/3)x^2$-3	-1; 1
20	$(e^x/2)$*sqrt(x)	2;3
21	ln(1+x)	1; 3
22	cos(x)·e^{2x}	1; 2

Примітка: ln(x) → log(x) в середовищі MatLanb

Аудиторна робота

Приклад 8.1. Порівняємо ефективність використання алгоритмів нечіткого виведення, які базуються на нечітких моделях Мамдані та Сугено 0-го порядку при розв'язанні задачі апроксимації неперервної функції від однієї змінної за критеріями забезпечення точності апроксимації та обчислювальних витрат на реалізацію. Порівняння описаних алгоритмів виконується при таких умовах: 1) апроксимація функції виконується на заданому відрізку; 2) функція, що апроксимується, задається набором значень (x_k, y_k), k = 1, ..., K, при цьому точки x_k розташовані на однаковій відстані; 3) ФН мають вигляд трикутника; 4) кількість правил n задано заздалегідь.

Апроксиамція функції $y=x^2$: нечіткі моделі Мамдани та Сугено 0-го порядку

Частина 1. Нечітка модель Мамдані

```
f_name='1_81'
x_t=16;
x_beg=1, x_end=4
n=x_t;
X=x_beg:((x_end-x_beg)/(n-1)):x_end;
delta_x=(x_end-x_beg)/(x_t-1)
n=n
for i=1:n
 Y(1,i)=X(1,i)^2;
end;
ymin=min(Y(1,:));ymax=max(Y(1,:));

 delta_y=(ymax-ymin)/x_t;
 figure(1)
```

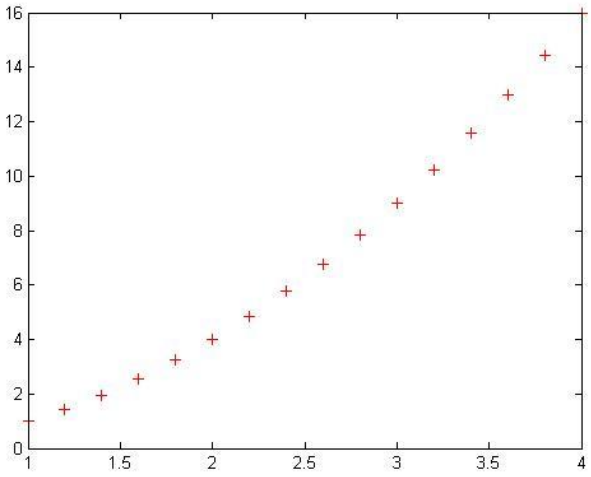
```
 xlabel('X'); ylabel('Y');
 axis ([x_beg-0.5 x_end+0.5 ymin-0.5 ymax+0.5]);
 plot(X(1,:),Y(1,:), '+r')

ai=newfis(f_name,'mamdani','min','max','min','max','centroid');
 ai=addvar(ai,'input', 'x', [x_beg x_end]);
 for i=1:n
   if i==1 a=x_beg-delta_x; else a=X(i-1)
   end
   b=X(i);
   if i==n c=x_end+delta_x; else c=X(i+1)
   end
  ai=addmf(ai,'input',1, ['x',int2str(i)],'trimf', [a b c]);
   end
   ai=addvar(ai,'output','y', [ymin ymax]);
   for i=1:n
   if i==1 a= ymin-delta_y; else a=Y(i-1)
   end
   b=Y(i);
   if i==n c=ymax+delta_y; else c=Y(i+1)
   end
   ai=addmf(ai,'output',1,['y',int2str(i)],'trimf',[a b c]);
 end
 rullist=ones(n,4);
 for i=1:n
   rullist(i,1)=i; rullist(i,2)=i;
 end
ai=addrule(ai,rullist);
writefis(ai,f_name);
a1=readfis(f_name);
a1=setfis(a1,'defuzzmethod','centroid');
outt1=zeros(1,n)
for i=1:n
 outt1(1,i)=evalfis(X(1,i),a1)
end
eps1=sum(abs(outt1(:)-Y'))/n
```

Частина 2. Нечітка модель Сугено

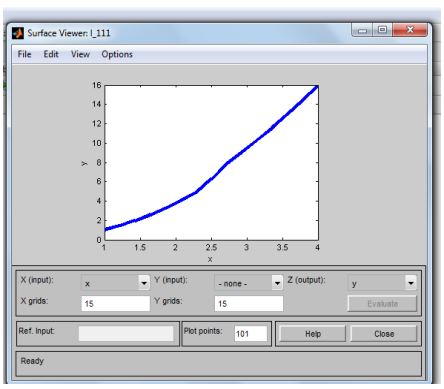
```
f_name='1_82'
x_t=16; x_beg=1, x_end=4
n=x_t; X=x_beg:((x_end-x_beg)/(n-1)):x_end;
X1=(x_beg+0.2):((x_end-x_beg)/(n-1)):x_end;
delta_x=(x_end-x_beg)/(x_t-1)
for i=1:n
 Y(1,i)=X(1,i)^2;
end;
ymin=min(Y(1,:));ymax=max(Y(1,:));delta_y=(ymax-ymin)/x_t;
ai=newfis(f_name,'sugeno','prod','probor','prod','sum','wtaver');
ai=addvar(ai,'input', 'x', [x_beg x_end]);
 for i=1:n
   if i==1 a=x_beg-delta_x; else a=X(i-1) end
   b=X(i);
```

```
   if i==n c=x_end+delta_x; else c=X(i+1) end
  ai=addmf(ai,'input',1, ['x',int2str(i)],'trimf', [a b c]);
   end
   ai=addvar(ai,'output','y', [1 n]);
   for i=1:n
   if i==1 a= ymin-delta_y; else a=Y(i-1); end
   b=Y(i);
   if i==n c=ymax+delta_y; else c=Y(i+1); end
 ai=addmf(ai,'output',1,['y', int2str(i)], 'constant', Y(i));
 end
 rullist=ones(n,4);
 for i=1:n
   rullist(i,1)=i; rullist(i,2)=i;
end
ai=addrule(ai,rullist); writefis(ai,f_name);
a2=readfis(f_name); outt2=zeros(1,n)
for i=1:n
 outt2(1,i)=evalfis(X(1,i),a2)
end
eps2=sum(abs(outt2(1,:)-Y(1,:)))/n
error=abs(eps1-eps2)
```

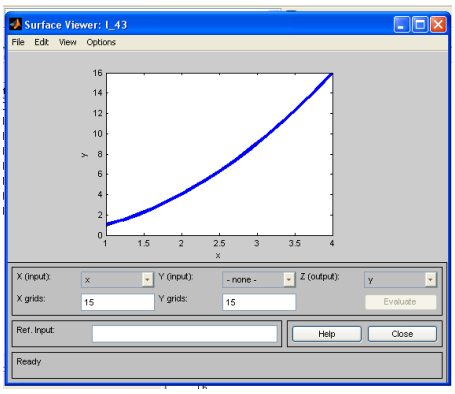
Результат



Графік реальної функції



а



б

Результ моделювання на основі СНВ: типу Мамдані (*а*) та Сугено (*б*)

№	x^k	y^k	Оцінка за алгоритмом	
			Мамдані (y)	Сугено (y)
1	1.0000	1.0000	1.0983	1.0000
2	1.2000	1.4400	1.4725	1.4400
3	1.4000	1.9600	1.9902	1.9600
4	1.6000	2.5600	2.5835	2.5600
5	1.8000	3.2400	3.2630	3.2400
6	2.0000	4.0000	4.0289	4.0000
7	2.2000	4.8400	4.8693	4.8400
8	2.4000	5.7600	5.7867	5.7600
9	2.6000	6.7600	6.7870	6.7600
10	2.8000	7.8400	7.8666	7.8400
11	3.0000	9.0000	9.0266	9.0000
12	3.2000	10.2400	10.2669	10.2400
13	3.4000	11.5600	11.5862	11.5600
14	3.6000	12.9600	12.9867	12.9600
15	3.8000	14.4400	14.4653	14.4400
16	4.0000	16.0000	15.5278	16.0000
			eps1 = 0.0594	eps2 = 4.8572e-016

9. Проектування нечітких правил на основі чисельних даних

Мета роботи: навчитися проектувати базу нечітких правил системи управління на основі чисельних даних. *Об'єкт дослідження* – нечітка система управління (алгоритм «Побудови бази правил на основі чисельних даних»).

Питання для опрацювання

1.Проектування бази нечітких правил на основі чисельних даних [13]

1. Завдання до практичної роботи

1. Розглянути і засвоїти процес проектування бази нечітких правил на основі чисельних даних.

Алгоритм «Побудови бази правил на основі чисельних даних»

Крок 1. *Поділ просторів змінних (сигналів) вхід-вихід на підобласті:* розділити простір значень x_1, x_2 і y на підобласті; сформувати відповідні ФН.

Крок 2. *Ініціалізація таблиць*: сформувати таблиці для фіксації 1) бази правил BR=0; 2) ступенів істинності правил $T[A_1, A_2]=0$. Встановити $i=1$.

Крок 3. Обрати точку даних ($x_1(i)$, $x_2(i)$, $y(i)$).

Крок 4. *Побудова нечітких правил на основі даних навчання*: визначити ступінь належності даних до областей (нечітких множин), сформувати відповідне правило вигляду: $R^{(i)}$: if $x_1 = A_1$ and $x_2 = A_2$ then $y = B$.

Крок 5. *Приписування кожному правилу значення ступеня істинності:* визначити ступінь істинності правила $R^{(i)}$ за формулою:

$SP(R^{(i)})=\mu_{A1}(x1)\ \mu_{A2}(x2)\cdot\mu_B(y)$.

Крок 6. Перевірити, чи $SP(R^{(i)}) > T[A_1, A_2]$? Якщо «так», то вписати правило $R^{(i)}$ в таблицю BR: $BR[A_1, A_2]=B$ і $T[A_1, A_2]= SP(R^{(i)})$, перейти до п. 7; інакше перейти до п. 7.

Крок 7. Встановити $i=i+1$

Крок 8. Перевірити, чи розглянуто всі пари даних: $i > n$? Якщо «так», то перейти до п. 9, інакше перейти до п. 3.

Крок 9. *Дефазифікація.* Зупинка алгоритму.

2. Розглянути і засвоїти процес розробки СНВ в системі MatLab в режимі команд: написати програму, яка формує базу нечітких правил.

2. Контрольні запитання

1. Описати алгоритм *«Побудови бази правил на основі чисельних даних»*.

Задачі для самостійного розв'язання: комп'ютерний практикум № 4.

Аудиторна робота

Приклад 9.1. Реалізувати алгоритм проектування бази нечітких правил на основі чисельних даних (даних навчання).

```
clc; clear all;
data=[[-100  180 45]
```

```
[-98.52  155.67 34.02]
[-92.08  133.22 28.56]
[-82.05  114.04 27.93]
[-70.21  95.24  23.05]
[-57.37  79.54  13.74]
[-44.23  70.01  13.18]
[-31.78  60.86  14.35]
[-20.47  50.92  23.09]
[-11.52  35.18  14.35]
[-4.59   25.01  15.71]
[0.09   14.14  14.37]
[2.41   4.19  10.58]
[2.66  -3.17   0.77]
[1.83  -3.71 -0.91]
[0.99  -3.07 -1.53]
[0.35  -2    -1.45]
[-0.03 -0.98 -1.07]];

x1_0=min(data(:,1));x1_n=max(data(:,1));
x2_0=min(data(:,2));x2_n=max(data(:,2));
y_0=min(data(:,3));y_n=max(data(:,3));
N=[5 7 5];
d_x1=(x1_n-x1_0)/(N(1)-1); d_x2=(x2_n-x2_0)/(N(2)-1);
d_y=(y_n-y_0)/(N(3)-1);
A=x1_0:d_x1:x1_n; B=x2_0:d_x2:x2_n;
y=y_0:d_y:y_n
% initialization------------------
a0=x1_0;an=x1_n;Na=N(1);da=d_x1;
b0=x2_0;bn=x2_n;Nb=N(2);db=d_x2;
y0=y_0;yn=y_n;Ny=N(3);dy=d_y;

[mf1]=mf_vx(a0,an,da,A); [mf2]=mf_vx(b0,bn,db,B);
[y_mf]=mf_vx(y0,yn,dy,y);
mf1,mf2,y_mf;

mf1_t=mf1';
rab1=zeros(Na+2,1); rab1=mf1_t(:,1);
r=length(data(:,1)); X1=zeros(r,1);X1=data(:,1);
[term_1,m1,Z1]=n_term(X1, rab1, r, Na)
mf2_t=mf2'; rab2=zeros(Nb+2,1); rab2=mf2_t(:,1);
X2=zeros(r,1);X2=data(:,2);
[term_2,m2,Z2]=n_term(X2, rab2, r, Nb)
ymf_t=y_mf';
rab_y=zeros(Ny+2,1); rab_y=ymf_t(:,1);
y=zeros(r,1);y=data(:,3);
[term_y,my,Zy]=n_term(y, rab_y, r, Ny)

rulles=zeros(Nb,Na);t_BR=zeros(Nb,Na);
for k=1:r
    k1=term_1(k); k2=term_2(k);
    if rulles(k2,k1)==0
       rulles(k2,k1)=m1(k)*m2(k)*my(k);
```

```
       t_BR(k2,k1)=term_y(k);
    else
       if rulles(k2,k1)<m1(k)*m2(k)*my(k)
           rulles(k2,k1)=m1(k)*m2(k)*my(k);
           t_BR(k2,k1)=term_y(k);
       end;
    end;
end;
rulles,t_BR
% deffuzifier
x=[-20 50]; % y=23
k=1;t1=zeros(2,2);t2=zeros(2,2);
for i=1:Na
    a=rab1(k,1);b=rab1(k+1,1);c=rab1(k+2,1);
    if ((x(1)>a && x(1)<b) || (x(1)>b && x(1)<c))
        t1(1,1)=i;t1(2,1)=trimf(x(1),[a b c]);
        a=rab1(k+1,1);b=rab1(k+2,1);c=rab1(k+3,1);
        t1(1,2)=i+1;t1(2,2)=trimf(x(1),[a b c]);
        break;
    end;
    k=k+1;
end;
t1
k=1;
for i=1:Nb
    a=rab2(k,1);b=rab2(k+1,1);c=rab2(k+2,1);
    if ((x(2)>a && x(2)<b) || (x(2)>b && x(2)<c))
        t2(1)=i;t2(2)=trimf(x(2),[a b c]);
        a=rab2(k+1,1);b=rab2(k+2,1);c=rab2(k+3,1);
        t2(1,2)=i+1;t2(2,2)=trimf(x(2),[a b c]);
        break;
    end;
    k=k+1;
end;
t2;
k=1;s1=0;s2=0;
for i=1:2
    for j=1:2
        tau(k)=t1(2,i)*t2(2,j); % tt(k)=t1(1,i)*t2(1,j)
        if t_BR(t1(1,i),t2(1,j))~=0

yy=(rab_y(t_BR(t1(1,i),t2(1,j))+1)+rab_y(t_BR(t1(1,i),t2(1,j))+2))
/2;
         s1=s1+tau(k)*yy; s2=s2+tau(k);
        end;
        k=k+1;
end; end;
y_output=s1/s2

function [U] = mf vx(a0,a1,d,y)
k=(a1-a0)/d+1;
zmf1=ones(k+1,k+2);
```

```
n=1;
zmf1(1,1)=a0; zmf1(1,k+2)=a1;
for i=a0:d:a1
    n=n+1;
    zmf1(1,n)=i;
end;
for j=1:length(y)
        if j==1;
            a=a0;
        else
            a=y(j-1);
        end
        b=y(j);
        if j==length(y)
            c=a1;
        else
            c=y(j+1);
        end
        zmf1(j+1,:)=trimf(zmf1(1,:),[a b c]);
end;
zmf1(2,1)=0; zmf1(k+1,k+2)=0;
U=zmf1;
end

function [term_1,m,Z1] = n_term(X1,rab,r,Na)
k=1; max_x=max(X1);
for i=1:Na
    a=rab(k),b=rab(k+1),c=rab(k+2);
    for j=1:r
        Z1(j,i)=trimf(X1(j), [a b c]);
        if (X1(j)==max_x) && (trimf(X1(j),[a b c])==0)
            Z1(j,Na)=1;
        end;
    end;
    k=k+1;
end;% Z1
m=zeros(r,1)
for i=1:Na
 for j=1:r
   m(j,1)= max(Z1(j,:));
end;end;
term_1=zeros(r,1)
for j=1:r
    for i=1:Na
       if m(j,1)==Z1(j,i)
            term_1(j,1)=i; break;
        end;
        if i==Na
            term_1(j,1)=0;
        end;
    end;
 end; end
```

Результат:

```
rab1 =                rab2 =                rab_y =
-  100.0000           -3.7100               -1.5300
 -100.0000            -3.7100               -1.5300
  -74.3350            26.9083               10.1025
  -48.6700            57.5267               21.7350
  -23.0050            88.1450               33.3675
    2.6600           118.7633               45.0000
    2.6600           149.3817               45.0000
                     180.0000
                     180.0000

t1 =   4.0000   5.0000                t2 =   2.0000   3.0000
       0.8829   0.1171                       0.2458   0.7542

y_output = 19.4225
t_BR =
   0   0   0   0   1
   0   0   0   2   2
   0   0   2   3   0
   0   3   2   0   0
   4   4   0   0   0
   4   0   0   0   0
   5   0   0   0   0
```

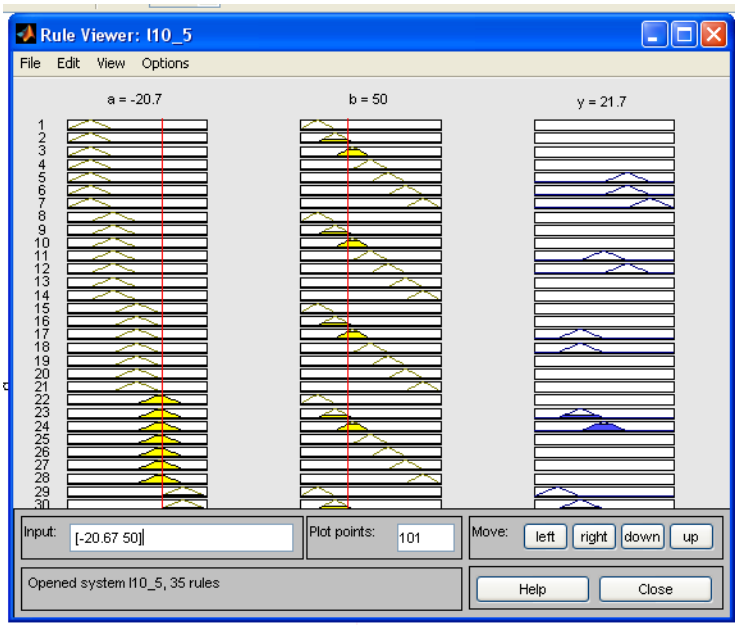
Приклад 9.2. Реалізувати алгоритм проектування бази нечітких правил на основі чисельних даних (даних навчання).

```
clc; clear all;
data=[[-100  180 45]
[-98.52  155.67 34.02]
[-92.08  133.22 28.56]
[-82.05  114.04 27.93]
[-70.21  95.24  23.05]
[-57.37  79.54  13.74]
[-44.23  70.01  13.18]
[-31.78  60.86  14.35]
[-20.47  50.92  23.09]
[-11.52  35.18  14.35]
[-4.59   25.01  15.71]
[0.09   14.14  14.37]
[2.41   4.19  10.58]
[2.66  -3.17   0.77]
[1.83  -3.71 -0.91]
[0.99  -3.07 -1.53]
[0.35  -2    -1.45]
[-0.03 -0.98 -1.07]];

x1_0=min(data(:,1));x1_n=max(data(:,1));
x2_0=min(data(:,2));x2_n=max(data(:,2));
y_0=min(data(:,3));y_n=max(data(:,3));
N=[5 7 5];
d_x1=(x1_n-x1_0)/(N(1)-1); d_x2=(x2_n-x2_0)/(N(2)-1);
```

```
d_y=(y_n-y_0)/(N(3)-1);
A=x1_0:d_x1:x1_n;B=x2_0:d_x2:x2_n;
y=y_0:d_y:y_n; f_name='l10_2'
% initialization------------------
a0=x1_0;an=x1_n;Na=N(1);da=d_x1;
b0=x2_0;bn=x2_n;Nb=N(2);db=d_x2;
y0=y_0;yn=y_n;Ny=N(3);dy=d_y;
ai=newfis(f_name, 'mamdani', 'min', 'max','min', 'max','mom');
%'centroid'
% adding var a
a0=x1_0;an=x1_n;Na=N(1);da=d_x1;
ai= addvar(ai,'input','a', [a0-da an+da]);
for i=1:Na
    if i==1
     a=a0-da;
    else
     a=A(i-1);
    end;
    b=A(i);
    if i==Na
     c=an+da;
    else
     c=A(i+1);
    end
   ai=addmf (ai,'input',1, ['a',int2str(i)],'trimf', [a b c]);
end;
% adding var b
b0=x2_0;bn=x2_n;Nb=N(2);db=d_x2;
ai= addvar(ai,'input','b', [b0-db bn+db]);
for i=1:Nb
    if i==1
     a=b0-db;
    else
     a=B(i-1);
    end;
    b=B(i);
    if i==Nb
     c=bn+db;
    else
     c=B(i+1);
    end
   ai=addmf (ai,'input',2, ['2',int2str(i)],'trimf', [a b c]);
end;
% adding var y
y0=y_0;yn=y_n;Ny=N(3);dy=d_y;
ai= addvar(ai,'output','y', [y0-dy yn+dy]);
for i=1:Ny
    if i==1
     a=y0-dy;
    else
     a=y(i-1);
    end;
```

```
    b=y(i);
    if i==Ny
     c=yn+dy;
    else
     c=y(i+1);
    end
  ai=addmf (ai,'output',1, ['y',int2str(i)],'trimf', [a b c]);
end;
a0=x1_0;an=x1_n;Na=N(1);da=d_x1;
b0=x2_0;bn=x2_n;Nb=N(2);db=d_x2;
y0=y_0;yn=y_n;Ny=N(3);dy=d_y;

[mf1]=mf_vx(a0,an,da,A); [mf2]=mf_vx(b0,bn,db,B);
[y_mf]=mf_vx(y0,yn,dy,y);
mf1,mf2,y_mf;

mf1_t=mf1';
rab1=zeros(Na+2,1); rab1=mf1_t(:,1);
r=length(data(:,1));X1=zeros(r,1);X1=data(:,1);
[term_1,m1,Z1]=n_term(X1, rab1, r, Na)

mf2_t=mf2';
rab2=zeros(Nb+2,1); rab2=mf2_t(:,1);
X2=zeros(r,1);X2=data(:,2)
[term_2,m2,Z2]=n_term(X2, rab2, r, Nb)

ymf_t=y_mf';
rab_y=zeros(Ny+2,1)
rab_y=ymf_t(:,1);
y=zeros(r,1);y=data(:,3)
[term_y,my,Zy]=n_term(y, rab_y, r, Ny)

rulles=zeros(Nb,Na);t_BR=zeros(Nb,Na);
for k=1:r
    k1=term_1(k); k2=term_2(k);
    if rulles(k2,k1)==0
       rulles(k2,k1)=m1(k)*m2(k)*my(k);
       t_BR(k2,k1)=term_y(k);
    else
       if rulles(k2,k1)<m1(k)*m2(k)*my(k)
           rulles(k2,k1)=m1(k)*m2(k)*my(k);
           t_BR(k2,k1)=term_y(k);
       end;
    end;
end;  %rulles,t_BR
rullist=ones(Na*Nb,5); s=0;
 for i=1:Na
   for j=1:Nb
       k=s+j;
       rullist(k,1)=i;rullist(k,2)=j;rullist(k,4)=0;
        rullist(k,3)=0;
       if t_BR(j,i)>0
```

```
        rullist(k,3)=t_BR(j,i); rullist(k,4)=1;
      end;
   end;
   s=s+j;
 end;
ai= addrule(ai, rullist); writefis (ai,f_name);
fuzzy 'l10_2'  %ai=readfis('f_name');
```



Теоретичні відомості

1. Проектування бази нечітких правил на основі чисельних даних [13]. При розв'язанні більшості прикладних задач регулювання інформацію, необхідну для побудови і реалізації системи управління, можна розділити на дві частини: чисельну (кількісну), одержувану з вимірювальних датчиків, і лінгвістичну (якісну), що надходить від експерта. Значна частина нечітких систем регулювання використовує другий вид знань, які найчастіше подаються у формі бази нечітких правил. Якщо виникає необхідність спроектувати нечітку систему при наявності тільки чисельних даних, ми стикаємося із проблемами.

Одним із шляхів їх вирішення вважаються так звані нейро-нечіткі (fuzzy-neural) системи. Вони мають багато переваг, проте стримуючим моментом є тривалість наповнення їх знаннями (побудови бази правил) в процесі ітеративного навчання. Далі розглянемо один з найпростіших, але досить універсальний метод побудови бази нечітких правил на основі чисельних

даних. Переваги цього методу полягають в його надзвичайній простоті і дуже високій ефективності. Крім того, він дозволяє об'єднувати чисельну інформацію, подану у формі навчальних даних, з лінгвістичною інформацією, що має вигляд бази правил, за рахунок доповнення наявної бази правилами, створеними на основі численних даних.

1. Побудова нечітких правил. Нехай ми створюємо базу правил для нечіткої системи з двома входами і одним виходом. Очевидно, що для цього необхідні навчальні дані у вигляді множини пар ($x1(i)$, $x2(i)$, $d(i)$), $i = 1, 2, \ldots$, де $x1(i)$, $x2(i)$ – сигнали, які подаються на вхід модуля нечіткого управління, а $d(i)$ - очікуване (еталонне) значення вихідного сигналу. Завдання полягає в формуванні таких нечітких правил, щоб сконструйований на їх основі модуль управління при отриманні вхідних сигналів генерував коректні (що мають найменшу похибку) вихідні сигнали.

Крок 1. *Поділ просторів вхідних і вихідних сигналів на області.* Нехай нам відомо мінімальне і максимальне значення кожного сигналу. За ним можна визначити інтервали, в яких знаходяться допустимі значення. Наприклад, для вхідного сигналу x_1 такий інтервал позначимо $[x_1^-, x_1^+]$. Якщо значення x_1^- і x_1^+ невідомі, то можна скористатися даними навчання і вибрати з них відповідно мінімальне і максимальне значення $x_1^- = \min(x1)$; $x_1^+ = \max(x1)$.

Аналогічно для сигналу x_2 визначимо інтервал $[x_2^-, x_2^+]$, а для еталонного сигналу d – інтервал $[d^-, d^+]$. Кожен визначений таким чином інтервал розділимо на $(2N+1)$ областей (відрізків), причому значення N для кожного сигналу підбирається індивідуально, а відрізки можуть мати однакову або різну довжину. Окремі області позначимо таким чином: MN (Малий N), ..., M1 (Малий 1), S (Середній), D1 (Великий 1), ..., DN (Великий N) і для кожного з них визначимо одну функцію належності. На рис. 3.37 зображено приклад такого поділу, де область визначення сигналу $x1$ розбито на п'ять підобластей ($N = 5$), сигналу $x2$ – на сім підобластей ($N=7$), тоді як область визначення вихідного сигналу y – на п'ять подобластей ($N = 5$).

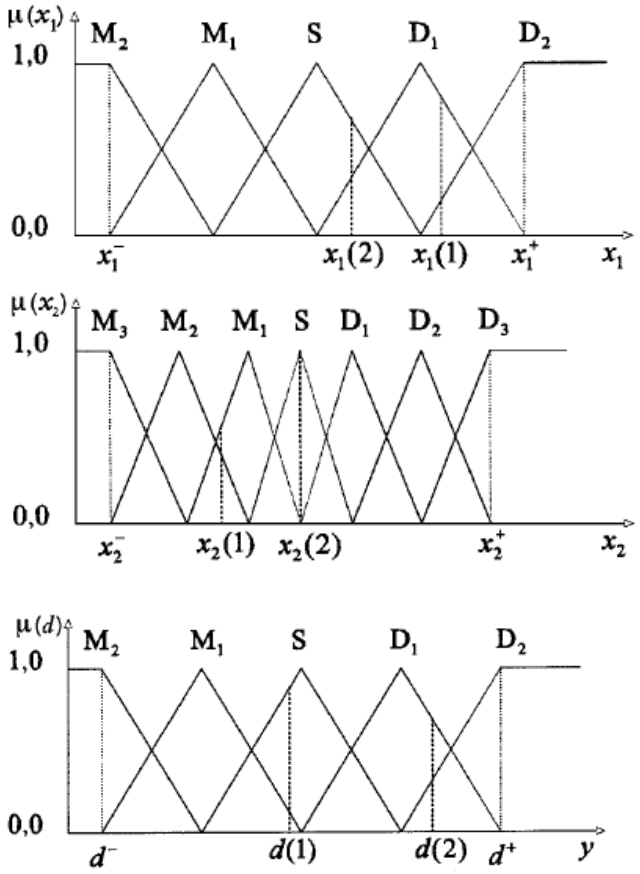


Рис. 10.1. Поділ просторів вхідних і вихідних сигналів на області і відповідні їм ФН

Кожна ФН має трикутну форму; одна з вершин розташовується в центрі області і їй відповідає значення функції, яке дорівнює «1». Дві інших вершини розташовані в центрах сусідніх областей, їм відповідають значення функції, які дорівнюють «0». Очевидно, що такий поділ вибрано для прикладу. Можна запропонувати багато інших способів поділу вхідного і вихідного простору на окремі області і використовувати інші форми ФН.

Крок 2. *Побудова нечітких правил на основі даних навчання.*. Спочатку визначимо ступінь належності навчальних даних ($x1(i)$, $x2(i)$, $d(i)$) до кожної області, виділеної на кроці 1. Ці ступені будуть виражатися значеннями ФН відповідних нечітких множин для кожної групи даних. Наприклад, для випадку, зображеному на рис. 3.37, ступінь належності значення $x1(1)$ до області D1 становить «0,8», до області D2 - «0,2», а до решти областей – «0». Аналогічно для даного $x2$ (2) ступінь належності до області S становить «1», а до решти

областей – «0». Тепер співставимо дані навчання $x1(i)$, $x2(i)$ і $d(i)$ до областей, в яких вони мають максимальні ступіні належності. Зауважимо, що $x1(1)$ має найбільшу ступінь належності до області D_1, а $x2$ (2) – до області S. Остаточно для кожної пари навчальних даних можна записати одне правило, тобто

$(x1(1), x2(1), d(1)) \Rightarrow$

{$x1(1)$ [max: 0,8 у D1], $x2(1)$ [max: 0,6 у M1]; $d1(1)$ [max: 0,9 у S]} $\rightarrow$

$R^{(1)}$: if (x_1 = D_1 and x_2 = M_1) then y = S;

$(x1(2), x2(2), d(2)) \Rightarrow$

{$x1(2)$ [max: 0,7 у S], $x2(2)$ [max: 1,0 у S]; $d1(2)$ [max: 0,7 у D1]} $\rightarrow$

$R^{(2)}$: if (x_1 = S and x_2 = S) then y = D_1;

Крок 3. *Приписування кожному правилу значення ступеня істинності*. Зазвичай, в наявності є велика кількість пар даних навчання, по кожній із них може бути сформульовано одне правило. Тому існує висока ймовірність того, що деякі з цих правил виявляться суперечливими. Це відноситься до правил з однією умовою, але з різними наслідками (висновками). Один з методів вирішення цієї проблеми полягає в приписуванні кожному правилу так званої ступеня істинності з подальшим вибором з суперечать один одному правил того, у якого ця ступінь виявиться найбільшою. Таким чином, вирішується проблема суперечливих правил і значно зменшується їх загальна кількість. Для правила вигляду «R: IF($x1$ = $A1$ AND $x2$ = $A2$) THEN(y=B)» ступінь істинності SP(R) визначають таким чином: $SP(R)=\mu_{A1}(x_1)\cdot\mu_{A2}(x_2)\cdot\mu_B(y)$.

Отже, перше правило $R^{(1)}$ з нашого прикладу має таку ступінь істинності:

$$SP(R^{(1)})=\mu_{D1}(x_1)\cdot\mu_{M1}(x_2)\cdot\mu_S(y)=0{,}8\cdot0{,}6\cdot0{,}9=0{,}432.$$

а друге правило: $SP(R^{(2)})=\mu_S(x_1)\cdot\mu_S(x_2)\cdot\mu_{D1}(y)=0{,}7\cdot1{,}0\cdot0{,}7=0{,}49$.

Крок 4. *Створення бази нечітких правил.* Спосіб побудови бази нечітких правил зображено на рис. 2. Ця база має вигляд таблиці, яка заповнюється нечіткими правилами таким чином: якщо правило має вигляд

R: IF($x1$ = $D1$ AND $x2$ = $M2$) THEN(y = S),

то на перетині рядка D1 (який відповідає сигналу $x1$) і стовпчика M1 (сигнал $x2$) вписуємо назву нечіткої множини, присутньої в наслідку, тобто S (яка відповідає вихідному сигналу y). За наявності кількох нечітких правил з однаковою умовою, з них вибирають правило, яке має найвищу ступінь істинності.

Крок 5. *Дефазифікація.* Завдання полягає у визначенні за допомогою бази правил відображення f: ($x1$, $x2$) $\rightarrow y$, де y – вихідна величина нечіткої системи. При визначенні кількісного значення керуючого впливу y для заданих вхідних сигналів ($x1$, $x2$) необхідно виконувати операцію дефаззифікації. Спочатку для вхідних сигналів ($x1$, $x2$) з використанням операції добутку об'єднаємо умови k-го нечіткого правила. Таким чином визначається ступінь активності k-го правила: $\tau^{(k)}=\mu_{A1}{}^{(k)}(x_1)\cdot\mu_{A2}{}^{(k)}(x_2)$. Наприклад, для першого правила $R^{(1)}$ ступінь активності визначають як $\tau^{(1)}=\mu_{D1}(x_1)\cdot\mu_{M1}(x_2)$. Для розрахунку вихідного значення y скористаємося методом центра ваги:

$$\overline{y}=\frac{\sum_{k=1}^{N}\tau^{(k)}\overline{y}^{(k)}}{\sum_{k=1}^{N}\tau^{(k)}}.$$

Розглянутий метод можна узагальнити на випадок нечіткої системи з будь-якою кількістю входів і виходів.

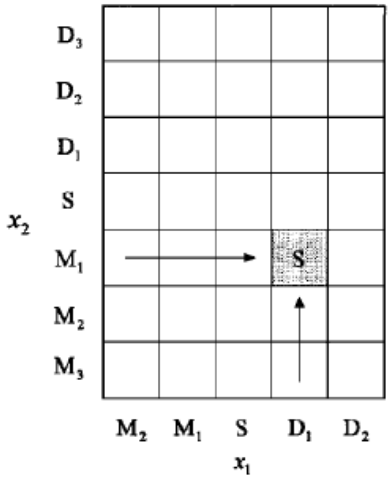


Рис. 10.2. Форма бази нечітких правил

Розділ 3. Деякі відомості з теорії нейронних мереж

Тема 1. Багатошаровий персептрон та алгоритм зворотного поширення похибки

10. Алгоритм зворотного поширення похибки

Мета роботи: ознайомитися із алгоритмом зворотного поширення похибки (як складової системи ANFIS). *Об'єкт дослідження*: алгоритм зворотного поширення похибки.

Питання для опрацювання

1. Алгоритм зворотного поширення похибки

1. Завдання до практичної роботи

1. Вивчити теоретичний матеріал.
2. Сформувати апроксиматор у вигляді двошарового персептрону на основі алгоритму зворотного поширення похибки.
3. Перевірити модель на здатність розв'язувати задачу апроксимації функції.

2. Контрольні запитання та завдання

1. Опишіть алгоритм зворотного поширення похибки.

Задачі для самостійного розв'язання: *комп'ютерний практикум* № 8

Аудиторна робота

Приклад 11.1. Нехай задано мережу (рис. 10.1), початкові значення ваги та зсуву якої мають такий вигляд: $w^1(0) = -1, b^1(0) = 1; w^2(0) = -2;$ $b^2(0) = 1$. На вхід мережі подається пара вхід–ціль {p = –1, t = 1}. Виконати одну ітерацію алгоритму зворотного поширення, якщо $\alpha = 1$.

Розв'язання. Спочатку обчислимо вихід мережі:

$$u^1 = w^1(0)p + b^1(0) = -1\cdot(-1) + 1 = 2;$$

$$y^1 = tansig\left(u^1\right) = \frac{\exp\left(u^1\right) - \exp\left(-u^1\right)}{\exp\left(u^1\right) + \exp\left(-u^1\right)} = \frac{\exp(2) - \exp(-2)}{\exp(2) + \exp(-2)} = 0,964;$$

$$u^2 = w^2(0)y^1 + b^2(0) = -2\cdot 0,964 + 1 = -0,928;$$

$$y^2 = tansig\left(u^2\right) = \frac{\exp\left(u^2\right) - \exp\left(-u^2\right)}{\exp\left(u^2\right) + \exp\left(-u^2\right)} = \frac{\exp(-0,928) - \exp(0,928)}{\exp(-0,928) + \exp(0,928)} = -0,7297;$$

$$e = \left(t - y^2\right) = 1 - \left(-0,7297\right) = 1,7297$$

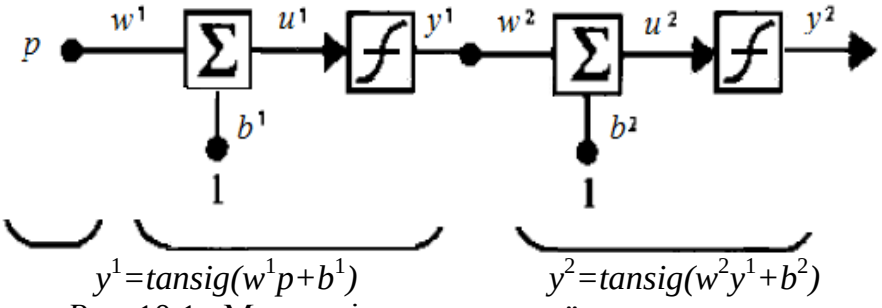


Рис. 10.1. Мережа із двома сигмоїдними шарами

Тепер визначимо коефіцієнти чутливості зворотного зв'язку для обох шарів:

– для другого:

$$s^2 = -2\dot{F}^2\left(u^2\right)(t-y) = -2\cdot\left(1-\left(y^2\right)^2\right)\cdot e = -2\cdot\left(1-\left(-0,7297\right)^2\right)\times 1,7297 = -1,6175;$$

– для першого:

$$s^1 = \dot{F}^1\left(u^1\right)w^2s^2 = \left(1-\left(y^1\right)^2\right)w^2s^2 = \left(1-\left(0,964\right)^2\right)\cdot(-2)\times\ \left(-1,6175\right) = 0,2285.$$

Після першої ітерації алгоритму зворотного поширення знайдемо оновлені значення ваги та зсуву:

$$w^2(1) = w^2(0) - \alpha s^2 y^1 = -2 - 1\cdot(-1,6175)\cdot 0,964 = -0,4407;$$

$$w^1(1) = w^1(0) - \alpha s^1 p = -1 - 1\cdot 0,2285\cdot(-1) = -0,7715;$$

$$b^2(1) = b^2(0) - \alpha s^2 = 1 - 1\cdot(-1,6175) = 2,6175;$$

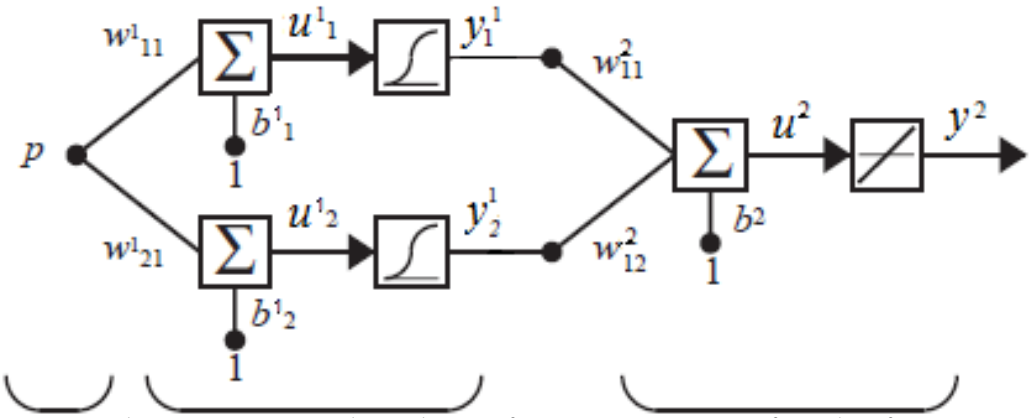
$$b^1(1) = b^1(0) - \alpha s^1 = 1 - 1\cdot(0,2285) = 0,7715.$$

Якщо ФА нейрона $y = f\left(u\right) = logsig(u) = \dfrac{1}{1+e^{-u}}$, то її похідну обчислюють за формулою $\dot{f}\left(u\right) = y(1-y)$. Якщо ФА нейрона $y = f\left(u\right) = tansig\left(u\right) = \dfrac{e^u - e^{-u}}{e^u + e^{-u}}$, то її похідну обчислюють за формулою

$$\dot{f}\left(u\right) = \frac{df\left(u\right)}{du} = \frac{d}{du}\left(\frac{e^u - e^{-u}}{e^u + e^{-u}}\right) = -\frac{e^u - e^{-u}}{\left(e^u + e^{-u}\right)^2}\left(e^u - e^{-u}\right) +$$

$$+\frac{e^u - e^{-u}}{e^u + e^{-u}} = 1 - \frac{\left(e^u - e^{-u}\right)^2}{\left(e^u + e^{-u}\right)^2} = 1 - \left(y\right)^2.$$

Приклад 10.2: MatLab. Реалізувати алгоритм зворотного поширення похибки: навчити мережу апроксимувати відображення вхід–вихід вигляду $y=p^2$ на основі двошарового персептрона зі структурою 1–2–1.



$\mathbf{y}^1 = \text{logsig}\ (\mathbf{w}^1 p + \mathbf{b}^1),\ y^2 = \text{purelin}\ ((\mathbf{w}^2)^T \mathbf{y}^1 + b^2)$

```
% function lab  - function approximation demonstration.
clc;clear all;close all;

s_1 = 2; % HIDDEN NEURONS
lr=0.1; p = -1:0.1:1;
T = p.^2; % FUNCTION APPROXIMATION
%========== Respond to train ================
 P2 = p;  xSize=size(p);
 errs=zeros(10000,1);  pp_rab=zeros(10000,1);
 S1 = s_1; R = 1; S2 = 1;
 W10 = rands (S1,1), B10 = rands (S1,1)
 W20 = rands (S1,1), B20 = rands (1,1);
 err_goal = 0.001 %1e-2 = 0.01
 epoch = 0; s_error=2;
 while (s_error > 1e-1 | epoch < 10000)
  s_error=0;
  for i=1:xSize(2)
   A0=p(i); u(:,1)=W10(:,1)*A0'+B10;
   for k=1:S1
    A1(k,1)=logsig(u(k));
   end;
   A2=(W20'*A1+B20); % purelin(W20'*A1+B20)
   e(i)=T(i)-A2; r=zeros(s_1,s_1);
   for k=1:s_1
    r(k,k)=(1-A1(k,1))*A1(k,1);
   end;
   ss2=(-2)*1*e(i); ss1=r*W20*ss2
   W21=W20-lr*ss2.*A1; B21=B20-lr*ss2;
   W11=W10-lr*ss1.*A0; B11=B10-lr*ss1;
   W10 = W11; B10 = B11;  W20 = W21; B20 = B21;
 end;
 epoch = epoch +1; s=0;
for i=1:xSize(2)
    s=s+abs(e(i));
 end
 s_error=s_error+s;
 pp_rab(epoch)=epoch;
```

```
 errs(epoch,1)=s_error;
 end % while
for i=1:xSize(2)
    A0=p(i);
    u(:,1)=W10(:,1)*A0'+B10;
    for k=1:S1
      A1(k,1)=logsig(u(k));
    end;
    A2=W20'*A1+B20;
    Z(i)=A2;
end;
 % INPUT GRAPH
 figure(); plot(pp_rab,errs);
 figure(); subplot(1,2,1)
 Target = plot(p,T,'*b','color',[0 0 1],'linewidth',3);
 title('Function'); xlabel('Input'); ylabel('Target');
 % OUTPUT GRAPH
 subplot(1,2,2)
 Attempt = plot(p,Z','Vr','color',[0 0 1],'linewidth',3);
 title('Approximation'); xlabel('Input'); ylabel('Target');

% function lab
clc;clear all;close all;
% Function approximation demonstration.
s_1 = 2; % HIDDEN NEURONS
lr=0.01;
p = -1:0.25:1; T = p.^2; % FUNCTION APPROXIMATION
%========== Respond to train ================
 P2 = p; rab=size(p)
 S1 = s_1; R = 1; S2 = 1;
 W10 = rands (S1,1), B10 = rands (S1,1)
 W20 = rands (S1,1), B20 = rands (1,1);
 err_goal = 0.001 %1e-2 = 0.01
 epoch = 0; s_error=2;
 while (abs(s_error)>1e-3 | (epoch<10000))
  s_error=0;
  for k=1:rab(2)
   A0=p(k); u(:,1)=W10(:,1)*A0+B10;
   for i=1:S1
    A1(i,1)=logsig(u(i));
   end;
   A2=(W20'*A1+B20); % purelin(W20'*A1+B20)
   e(k)=T(k)-A2; r=zeros(s_1,s_1);
   for i=1:s_1
    r(i,i)=(1-A1(i,1))*A1(i,1);
   end;
   ss2=(-2)*1*e(k); ss1=r*W20*ss2
   W21=W20-lr*ss2*A1; B21=B20-lr*ss2;
   W11=W10-lr*ss1*A0; B11=B10-lr*ss1;
   W10 = W11; B10 = B11;  W20 = W21; B20 = B21;
 end;
 epoch = epoch +1;
```

```
 s_error=s_error+sum(e(:));
 end % while
s_error,epoch; Y=zeros(rab(2),1);
for k=1:rab(2)
 pp=p(k);
for i=1:S1
  u=W10(i,1)*pp+B10; A1(i,1)=logsig(u(i));
end;
 Y(k,1)=W20'*A1+B20;
end;
 % INPUT GRAPH
 subplot(1,2,1)
 Target = plot(p,T,'*b','color',[0 0 1],'linewidth',3);
 title('Function'); xlabel('Input'); ylabel('Target');
 % OUTPUT GRAPH
 subplot(1,2,2)
 Attempt = plot(p,Y','Vr','color',[0 0 1],'linewidth',3);
 title('Approximation'); xlabel('Input'); ylabel('Target');
```

Результат

1. Апроксимація функції $y=p^2$ з кроком 0.25, s_1 = 2 – нейрони прихованого шару

p = -1:0.25:1; T = p.^2; s_error = 1.6190e-005; epoch = 5000

W21 =[3.0047; 2.6754] B11 =[-2.8984; -2.8657]

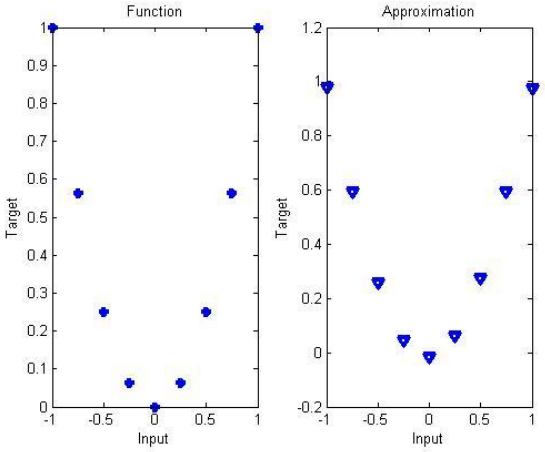
W11 =[-2.6104; 2.7853] B21 = -0.2977

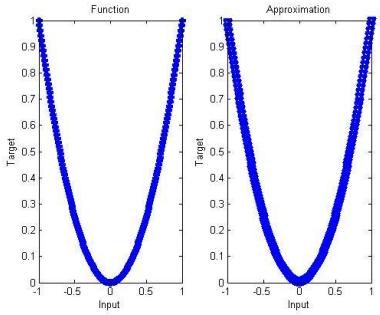
2. Апроксимація функції $y=p^2$ з кроком 0.01

p = -1:0.01:1; T = p.^2; s_error = -2.4766e-005; epoch = 5000

W21 =[2.9310; 2.6836]; B11 =[-2.9161; -2.8711]

W11 =[2.6606; -2.7668]; B21 = -0.2899





0.25 0.01

$\mathbf{y}^1=\text{tansig}(\mathbf{w}^1p+\mathbf{b}^1)$, $y^2=\text{tansig}\,((\mathbf{w}^2)^{\text{T}}\mathbf{y}^1+b^2)$

Результат роботи

А. epoch = 932388; s_error = 0.0883;
W10 =[8.2031; 0.4510]; B10 =[8.7864; -0.9741];
W20 =[-5.6686; -4.5198]; B20 = 3.6035;
Б. epoch = 924000; s_error = 0.0920; lr=0.01;

$$\mathbf{y}^1=\text{tansig}(\mathbf{w}^1 p+\mathbf{b}^1),\ y^2=\text{purelin}\ ((\mathbf{w}^2)^T\mathbf{y}^1+b^2)$$

А. epoch = 100000; s_error = 0.0137; lr=0.01;
epoch = 930000; s_error = 0.0138
Б. epoch = 930000; s_error = 0.0142; lr=0.001;

$$\mathbf{y}^1=\text{logsig}\ (\mathbf{w}^1 p+\mathbf{b}^1),\ y^2=\text{logsig}\ ((\mathbf{w}^2)^T\mathbf{y}^1+b^2)$$

s_error = -9.9618e-004; epoch = 17008; lr=0.01;
e = [0.1290 -0.0504 0.0146 -0.0332 -0.0610]^T

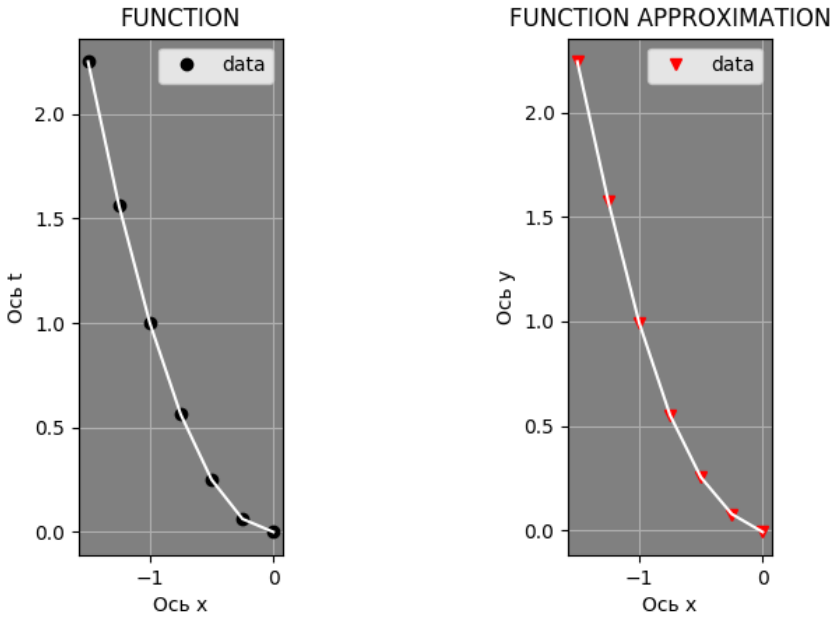
Лістинг Python

```
# function lab5 - FUNCTION APPROXIMATION
from math import exp
import itertools
from matplotlib import pyplot as plt
from matplotlib import rcParams
import random

def logsig(x):
    y=exp(-x);  y=1/(1+y)
    return (y)
def mult v1v2(V1,V2):
    ll=len(V1); s=0
    for i in range(0,ll):
      s+=V1[i]*V2[i]
    return (s)

# NEURONS
s_1 = 2;  # HIDDEN NEURONS
S1=s_1; S2=1
p=[];t=[] # input-output
lr=0.01;
for i in itertools.count(start=-1.5, step=0.25):
  if i>0: break;
  p.append(i); t.append(i**2)
#=====================================================
# Respond to train
#=====================================================
rab=len(p); W10=[]; B10=[]; W20=[]
for i in range(0,S1):
  arr=random.uniform(-1,1) # випадкове дійсне число від -1 до1
  W10.append(arr); arr=random.uniform(-1,1)
  B10.append(arr); W20.append(arr)
B20=random.uniform(-1,1)
err_goal = 0.01 # 1e-2=0.01
epoch = 0; s_error=2; e=[]
```

```
while (epoch<500000): # (s_error)>1e-2
  s_error=0
  for k in range(0,rab):
    A0=p[k];A1=[]
    for i in range(0,S1):
      arr=W10[i]*A0+B10[i];
      A1.append(logsig(arr))
  # print(A1)
    A2=mult_v1v2(W20,A1)+B20     # purelin()
    if epoch==0:
        e.append(t[k]-A2)
    else:
        e[k]=t[k]-A2
    r=[]
    for i in range(0,S1):
       r.append([0,0])  # range(0,S1)
    for i in range(0,S1):
       r[i][i]=(1-A1[i])*A1[i]
    ss2=(-2)*1*e[k];  ss1=[]
    for i in range(0,S1):
      arr=r[i][i]*W20[i]*ss2; ss1.append(arr)
    W21=[]
    for i in range(0,S1):
      arr=W20[i]-lr*ss2*A1[i]; W21.append(arr)
    B21=B20-lr*ss2;
    W11=[];B11=[]
    for i in range(0,S1):
      arr=W10[i]-lr*ss1[i]*A0; W11.append(arr)
      arr=B10[i]-lr*ss1[i]; B11.append(arr)
    W10=W11; B10=B11; W20=W21; B20=B21;
  epoch+=1
  s_error+=sum(e[0:rab])
print(s_error,epoch); y=[]
for k in range(0,rab):
    pp=p[k]; A1=[];u=[]
    for i in range(0,S1):
       arr=W10[i]*pp+B10[i]; u.append(arr)
       arr=logsig(u[i]); A1.append(arr)
    y.append(mult_v1v2(W20,A1)+B20)
fig=plt.figure();
ax1=fig.add_subplot(131, polar=False)
rect = ax1.patch; rect.set_facecolor('gray')
ax1.plot(p,t,'ok'); ax1.legend(['data'], loc='best')
ax1.plot(p,t,'w'); ax1.set_title(u'FUNCTION')
ax1.set_xlabel(u'Ось x');ax1.set_ylabel(u'Ось t')
ax1.grid(True)
ax2=fig.add_subplot(133, polar=False)
rect = ax2.patch; rect.set_facecolor('gray')
ax2.plot(p,y,'vr');ax2.legend(['data'], loc='best')
ax2.plot(p,y,'w'); ax2.set_title(u'FUNCTION APPROXIMATION')
ax2.set_xlabel(u'Ось x'); ax2.set_ylabel(u'Ось y'); ax2.grid(True)
plt.show()
```

Результат

Похибка апроксимації: 1.07410e-05



Приклад 10.3: python. Апроксимація функції вигляду $f(x)=x\cdot\sin(x)$, $x\in[0;\ 2\pi]$, якщо кількість точок апроксимації дорівнює значенню «5». Мережа має структуру 5-3-5 (пакетний режим навчання).

```
import numpy as np
import matplotlib.pyplot as plt

def train(x, y, size, n_h=3, num_it=150, rate=0.1):
    m = x.shape[0];  costs = []
    w1 = np.random.rand(size, n_h)
    w2 = np.random.rand(n_h, size)
    for i in range(num_it):
        z1 = np.dot(x, w1); a1 = np.tanh(z1)
        z2 = np.dot(a1, w2)
        cost = np.sum(np.power((y - z2), 2))/(2*size)
        costs.append(cost)
        dz2 = (y - z2)/size
        dw2 = 1/m*(np.dot(a1.T, dz2))
        dz1 = np.dot(dz2, w2.T) * (1 - np.power(a1, 2))
        dw1 = 1/m*(np.dot(x.T, dz1))
        w1 += rate * dw1; w2 += rate * dw2
  # plt.plot(costs, marker='o'); plt.show()
    params = {}
    params['w1'] = w1; params['w2'] = w2
    return params
def predict(x, parameters,size):
    w1 = parameters['w1']; w2 = parameters['w2']
    z1 = np.dot(x, w1);  a1 = np.tanh(z1)
    z2 = np.dot(a1, w2)
    for i in range(size):
        print("{}: -> {}".format(x[0][i], z2[0][i]))
```

```
    return z2
def func(x): return x*np.sin(x) # функція апроксимації

def main():  # x*sin(x) [0; pi]
    PI = 3.14; size = 5
    x_tr=np.array([(i*2*PI/size)    for    i    in    range(size)])
[np.newaxis]
    y_tr = np.array([func(x) for x in x_tr])
    parameters = train(x_tr, y_tr, size)
    print ("parameters=", parameters)
    y_pred = predict(x_tr, parameters,size)
    fig = plt.figure()
    ax1 = fig.add_subplot(131, polar=False)
    rect = ax1.patch; rect.set_facecolor('gray')
    ax1.plot(x_tr, y_tr, 'ok')
    ax1.legend(['data'], loc='best')
    ax1.plot(x_tr[0], y_tr[0], 'w')
    ax1.set_title(u'FUNCTION')
    ax1.set_xlabel(u'Ось x'); ax1.set_ylabel(u'Ось t')
    ax1.grid(True)
    ax2 = fig.add_subplot(133, polar=False)
    rect = ax2.patch
    rect.set_facecolor('gray')
    ax2.plot(x_tr, y_pred, 'vr')
    ax2.legend(['data'], loc='best')
    ax2.plot(x_tr[0], y_pred[0], 'w')
    ax2.set_title(u'FUNCTION APPROXIMATION')
    ax2.set_xlabel(u'Ось x'); ax2.set_ylabel(u'Ось y')
    ax2.grid(True); plt.show()

if __name__ == '__main__':
    main()
```

Результат

```
{'w1': array([[ 0.96177084,  0.60817625,  0.61231984],
    [ 0.17256333,  0.50888775,  0.32523658],
    [ 0.53036238,  0.26043479,  0.85034794],
    [ 0.97889787,  0.62465039,  0.95973182],
    [ 0.0525584 ,  0.27919627,  0.03605837]]),

'w2': array([[ 0.38743049,  0.55104382,  0.79524229, -0.9258566 , -1.24046911],
    [-0.28276332,  0.23447785,  0.45589307, -0.67071216, -1.6138187 ],
    [-0.10459511,  0.40888555,  0.22798958, -0.61208977, -1.92737718]])}
```

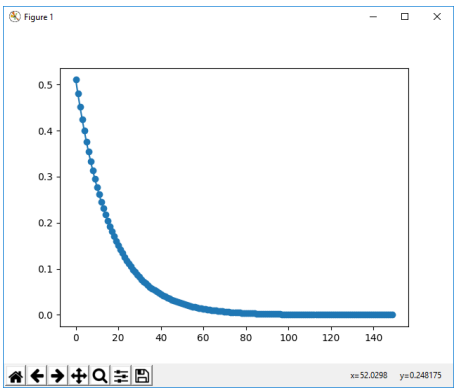


Рис. 10.2. Графік зміни значення похибки

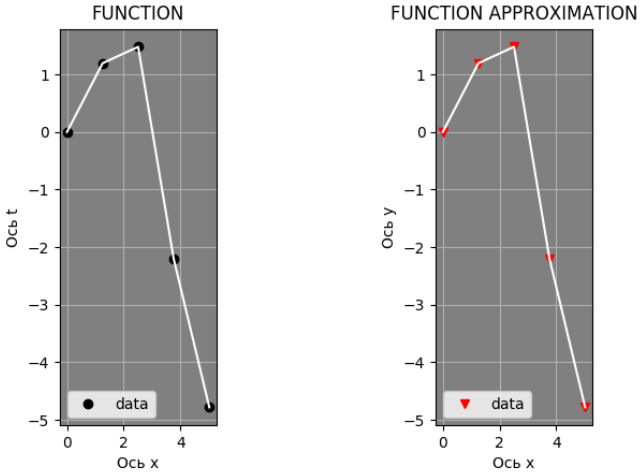


Рис. 10.3. Результат апроксимації

Теоретичний матеріал

1. Алгоритм зворотного поширення похибки [5, с. 156–174]. У системах контролю зазвичай постає проблема визначення функції апроксимації, яка встановлює відповідність між входом і виходом. У розглянутому далі прикл. 10.4 проілюстровано ефективність застосування нейронної мережі у вигляді багатошаровий персептрон (БП) для апроксимації функції $y=g(p)$ на основі двошарової мережі зі структурою 1–2–1, функції активації (ФА) якої мають вигляд $f^1(u) = logsig(u) = \frac{1}{1+e^{-u}}$ та $f^2(u) = purelin(u) = u$. Двошарові мережі із сигмоїдною ФА у прихованому шарі та лінійною ФА у вихідному шарі можуть апроксимувати будь-яку функцію з будь-якою точністю.

Розглянемо алгоритм навчання БП у вигляді *алгоритму зворотного поширення*.

Алгоритм зворотного поширення складається з таких етапів.

Крок 1. Ініціалізація ваги. Зазвичай початкове значення ваги (як і зсуву) дорівнює випадковим числам, рівномірно розподіленим на відрізках $[-1/2R, 1/2R]$ для першого шару та $[-1/2S^i, 1/2S^i]$ для $(i + 1)$-го шару (де $\mathbf{p} \in \Re^R$, S^i – кількість нейронів у i-му шарі). Зсув використовують

як значення ваги, що відповідає з'єднанню з фіктивним входом, який завжди дорівнює одиниці.

Крок 2. Прямий прохід. На входи всіх шарів мережі подаються відповідні значення:

$$\mathbf{y}^0 = \mathbf{p}; \mathbf{y}^{m+1} = \mathbf{f}^{m+1}\left(\mathbf{W}^{m+1}\mathbf{y}^m + \mathbf{b}^{m+1}\right), m = 0,\ 1,\ \ldots,\ M-1;\ \mathbf{y} = \mathbf{y}^M.$$

Крок 3. Зворотний прохід. Зворотне поширення похибки вздовж всієї мережі у вигляді чутливості функції мережі $F(\mathbf{x})$ до змін елементів вагового вектора:

$$\mathbf{s}^M = -2\dot{\mathbf{F}}^M\left(\mathbf{u}^M\right)(\mathbf{t}-\mathbf{y});\ \ \mathbf{s}^m = \dot{\mathbf{F}}^m\left(\mathbf{u}^m\right)\left(\mathbf{W}^{m+1}\right)^{\mathrm{T}}\mathbf{s}^{m+1},\ m = M-1, \ldots, 2, 1.$$

Крок 4. Налаштування значень ваги та зсуву за допомогою правила найшвидшого спуску вигляду

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha\mathbf{s}^m\left(\mathbf{y}^{m-1}\right)^{\mathrm{T}};\ \mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha\mathbf{s}^m. \quad (10.1)$$

Крок 5. Перевірка виконання критерію зупинки алгортму: значення $F(\mathbf{x})$, де $\mathbf{x} = [\mathbf{w}\ \ \mathbf{b}]^{\mathrm{T}}$, не змінюється на множині навчання (або різниця між виходом мережі та бажаним виходом t на всій множині навчання досягає деякого прийнятного значення). Якщо критерій виконується, то виконання алгоритму закінчують, інакше змінюють значення входу $\mathbf{p}$ і переходять до кроку 2.

Якщо мережа не збігається до розв'язку, то може бути потрібна більша або менша кількість нейронів прихованого шару. Вага деяких нейронів рідко змінюється, тому ці зв'язки (або нейрони) можуть не брати участі у процесі навчання – їх можна видалити, і кількість прихованих нейронів виявиться значно меншою.

Тема 2. РБФ–мережі

11. Розробка РБФ–мереж: гібридний алгоритм навчання

Мета роботи: ознайомитися з принципами функціонування та навчання РБФ–мережі. *Об'єкт дослідження*: РБФ–мережа, її використання для розв'язання задач класифікації та апроксимації.

Питання для опрацювання

1. Нейронні мережі на основі радіальних базисних функцій Гаусса
2. Гібридний алгоритм навчання ГРБФ-мереж [5].

1. Завдання до практичної роботи

1. Вивчити теоретичний матеріал.

2. Побудувати та навчити РБФ–мережу розпізнавати об'єкти, подані у вигляді точок одновимірного простору.

Спроектувати апроксиматор у вигляді РБФ–мережі (реалізувати алгоритм навчання), дослідити стійкість мережі.

Перевірити мережу на здатність розв'язувати задачу апроксимації функції.

2. Контрольні запитання та завдання

1. Опишіть структуру і принцип функціонування РБФ–мережі.

Задачі для самостійного розв'язання: комп'ютерний практикум № 8

Аудиторна робота

Приклад 11.1: MatLab. Спроектувати РБФ–мережу Гауса, яка апроксимує функцію y=2x+5 у трьох точках на інтервалі [–1 1], якщо S1=3 (кількість елементів у прихованому шарі).

Розв'язання:

1. **Звичайний метод**

Маємо точки $p_i \in \{-1, 0, 1\}$. Значення $c_i = p_i$, σ визначимо за формулою $\sigma = d_{max}/\text{sqrt}(2S^1) = (1-(-1))/\text{sqrt}(6)$. Формуємо матрицю **G**, визначаємо вектор ваги $\mathbf{w} = \mathbf{G}^{+}\mathbf{t} = (\mathbf{G}^{T}\mathbf{G})^{-1}\mathbf{G}^{T}$.

```
clc; clear all; close all; S1=3; npts=3; range=[-1 1];
d1=(range(2)-range(1))/(npts-1);p=range(1):d1:range(2);
t=2*p+5;t=t'; sigma=(range(2)-range(1))/sqrt(2*S1);
for i1=1:length(p)
 c(i1)=p(i1);
 for i2=1:length(p)
   u(i1,i2)=(p(i2)-c(i1))/sigma;
   G(i1,i2)=exp(-0.5*u(i1,i2).^2);
 end; end;
w=inv(G'*G)*G'*t;
y=G*w
```

Результат моделювання

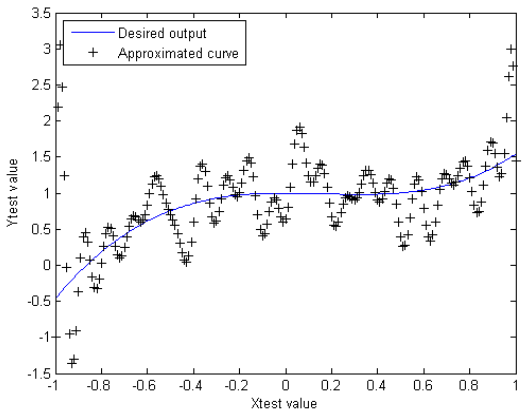
w = 2.2665 y = 3.0000 p = -1
0.8703 5.0000 0
6.4760 7.0000 1

G =

1.0000 0.4724 0.0498
0.4724 1.0000 0.4724
0.0498 0.4724 1.0000

2. **Звичайний метод із використанням шуму:** $y=x^3+\cos(x)$ на інтервалі [–1 1],

```
script % RBF1per
clc;clear all;close all;
% початкові дані
x=-1:0.05:1 % training datawith random noise
for i=1:length(x)
 y(i)=(x(i).^3+cos(x(i))+0.3*randn);
 end; t=length(x);
 for i=1:1:t
   for j=1:1:t
    h=x(i)-x(j); k =h^2/.02;
    train(i,j)=exp(-k);
 end;end; % data
W=inv(train)*y'; % testing the trained RBF
xtest=-1:0.01:1 % ytest is the desired output
ytest=xtest.^3+cos(xtest); % the interpolation matrix for test
t1=length(xtest); t= length(x);
for i=1:1:t1
   for j=1:1:t
    h=xtest(i)-x(j);
    k=h^2/.02;     test(i,j)=exp(-k);
 end;end; % data
actual_test=test*W;
plot(xtest,ytest,'b-', xtest,actual_test,'k+');
xlabel('Xtest value'); ylabel('Ytest value');
h=legend('Desired output', 'Approximated curve', 2);
set(h)
```



3. **Метод із використанням сингулярного розкладання.**

Сингулярне розкладання (Singular Value Decomposition, SVD). Для будь-якої невиродженої матриці **G** розмірності $N \times S^1$ сингулярне розкладання задається таким чином: $\mathbf{G} = \mathbf{U}\Lambda\mathbf{V}^{\mathbf{T}}$, де **U** – матриця розмірності $[N \times S^1]$, стовпці якої є власними векторами матриці $\mathbf{G}\mathbf{G}^{\mathrm{T}}$; **Λ** – діагональна матриця розмірності $[S^1 \times S^1]$, елементи якої називають сингулярними, і вони мають вигляд квадратного кореня із власних значень матриці $\mathbf{G}^{\mathrm{T}}\mathbf{G}$; **V** – матриця розмірності $[S^1 \times S^1]$, стовпці якої є власними векторами матриці $(\mathbf{G}^{\mathrm{T}}\mathbf{G})$. Матриці **U** та **V** – ортонормовані, тобто $\mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{I}$, $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}$. Підставляючи (9.6) у рівняння $(\mathbf{G}^{\mathrm{T}}\mathbf{t}) = (\mathbf{G}^{\mathrm{T}}\mathbf{G}\mathbf{w})$ отримаємо: $\mathbf{V}\Lambda\mathbf{U}^{\mathrm{T}}\mathbf{U}\Lambda\mathbf{V}^{\mathrm{T}}\mathbf{w} = \mathbf{V}\Lambda\mathbf{U}^{\mathbf{T}}\mathbf{t}$. Використовуючи рівняння $\mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}$, ваговий вектор можна визначити за формулою: $\mathbf{w} = \mathbf{V}\Lambda^{-1}\mathbf{U}^{\mathrm{T}}\mathbf{t} = \sum_{i=1}^{S^1} \mathbf{v}_i \left(\frac{\mathbf{u}_i^T \mathbf{t}}{\lambda_i} \right)$, $\lambda_i \neq 0$, де $\mathbf{u}_i$ та $\mathbf{v}_i$ – *i*-ий вектор-стовпець матриць **U** та **V** відповідно; λ_i – *i*-е сингулярне значення (діагональний елемент матриці **Λ**).

```
clc; clear all; close all;
S1=3; npts=3; range=[-1 1]; d1=(range(2)-range(1))/(npts-1)
p=range(1):d1:range(2); t=2*p+5;t=t';
sigma=(range(2)-range(1))/sqrt(2*S1);
for i1=1:length(p)
 c(i1)=p(i1);
 for i2=1:length(p)
   u(i1,i2)=(p(i2)-c(i1))/sigma;
   G(i1,i2)=exp(-0.5*u(i1,i2).^2);
 end; end;
f1=G'*G; f2=G*G'; [M1,M2]=eig(f1); [N1,N2]=eig(f2);
lam=sqrt(M2); w=M1*inv(lam)*N1'*t
```

Результат моделювання

w =	y =	p =
2.2665	3.0000	-1
0.8703	5.0000	0
6.4760	7.0000	1

Приклад 11.2: MatLab. *Гібридний алгоритм навчання РБФ–мереж.* Використовуючи гібридний алгоритм навчання, спроектувати ГРБФ, яка апроксимує функцію *y*=sin(2pi·(freq·*p*+phase/360))+1 на інтервалі [–1; 1], якщо кількість елементів у прихованому шарі S1=3.

```
clc; clear all; close all; % RBF_lab - FUNCTION APPROXIMATION
S1 = 3; % HIDDEN NEURONS (min = 2)
npts = 12;  % Number of Points: min = 2, max = 20
freq = 1/2;
% frequency of the function to be fit min = 0.25 max=1
```

```
phase = 60;  % phase of the function min = 0 max = 360
range = [-1 1]; % область визначення входу
d1 = (range(2)-range(1))/(npts-1);chag=d1
p = range(1):d1:range(2);
T = sin(2*pi*(freq*p+phase/360))+1;T=T';
lr=0.02;
%=====================================================
% Respond to train
%=====================================================
  rab=size(p); i=1
  while (i<=S1)
   c(i)=p(i); sigma(i)=3*chag; i=i+1;
  end
  %c;sigma;
  R=1; S2=1;
% for k1=1:rab(2) w(k1)=rands(1);  end
  w = rands(S1,1); err_goal = 0.001 %1e-3
 epoch=0;
 while  (epoch<10^6 || s>1e-3)
 s=0;u=zeros(rab(2),S1);
 for k1=1:rab(2)
  e=0;
  for k2=1:S1
   rab_e=((p(k1)-c(k2))^2/(sigma(k2)^2));
   u(k1,k2)=u(k1,k2)+ rab_e;
   f(k1,k2)=exp(-0.5*u(k1,k2)); e=e+w(k2)*f(k1,k2);
end;
 y(k1)=e; s=s+(e-T(k1,1))^2;
end;
 s=s/rab(2); epoch=epoch+1;
 if s<err_goal
   break;  end;
 if epoch>10^6
   break;
 end;
 de_dc=zeros(S1);  de_dsig=zeros(S1);  de_dw=zeros(S1);
 for k2=1:S1
  for k1=1:rab(2)
   rab_e=y(k1)-T(k1,1);
   de_dc(k2)=de_dc(k2)+rab_e*w(k2)*exp(-0.5*u(k1,k2))*((p(k1)-
c(k2))/(sigma(k2)^2));
   de_dsig(k2)=de_dsig(k2)+rab_e*w(k2)*exp(-0.5*u(k1,k2))*((p(k1)-
c(k2))^2 /(sigma(k2)^3));
   de_dw(k2)=de_dw(k2)+ rab_e*exp(-0.5*u(k1,k2));
 end;end;
 for k1=1:S1
   c(k1)=c(k1)-lr*de_dc(k1);
   sigma(k1)=sigma(k1)-lr*de_dsig(k1);
   w(k1)=w(k1)-lr*de_dw(k1);
 end;
end; % while
c, sigma, w
```

```
Y=f*w; % Y=w'*f'
E=(Y'-T').^2; RES=sum(E(:))/rab(2);
% GRAPH1
figure(1)
   subplot(1,2,1), Target =  plot(p,T,'*k','linewidth',1);
   title('Function'); xlabel('Input'); ylabel('Target');
   subplot(1,2,2)
   Attempt = plot(p,T,'*k',p,Y','ok','linewidth',1);
   title('Approximation - o');
   xlabel('Input'); ylabel('Target - *');
   total = range(2)-range(1);
   % вектор p2 (множина тестування)
   p2 = range(1):(total/25):range(2);  rab1=size(p2);
   yy=zeros(S1,rab1(2));
   for k1=1:S1
 for i=1:rab1(2)
   yy(k1,i)=exp(-0.5*(p2(i)-c(k1))^2/(sigma(k1)^2))
 end;   end;
   Y1=w'*yy
   for i=1:rab1(2)
   T1(i) = sin(2*pi*(freq*p2(i)+phase/360))+1;
   end
   Y1=w'*yy;RES1=sum((Y1-T1).^2)/rab1(2)
  % GRAPH2
figure(2); subplot(2,1,1)
for k1=1:S1
  Attempt=plot(p2(:),yy(k1,:),'-k','linewidth',1); hold on
end
ylabel('mu');  xlabel('Input 1'); hold off
subplot(2,1,2)
Attempt = plot(p2,T1,'*k',p2,Y1','ok','linewidth',1);
title('Approximation - o'); xlabel('Input1');
ylabel('Target - *')
```

Результат моделювання

Апроксимація функції sin(2pi·(freq·p+phase/360))+1

RES = 9.9991e-004 – на множині навчання:

p = -1.0000 -0.8182 -0.6364 -0.4545 -0.2727 -0.0909 0.0909 0.2727 0.4545 0.6364 0.8182 1.0000

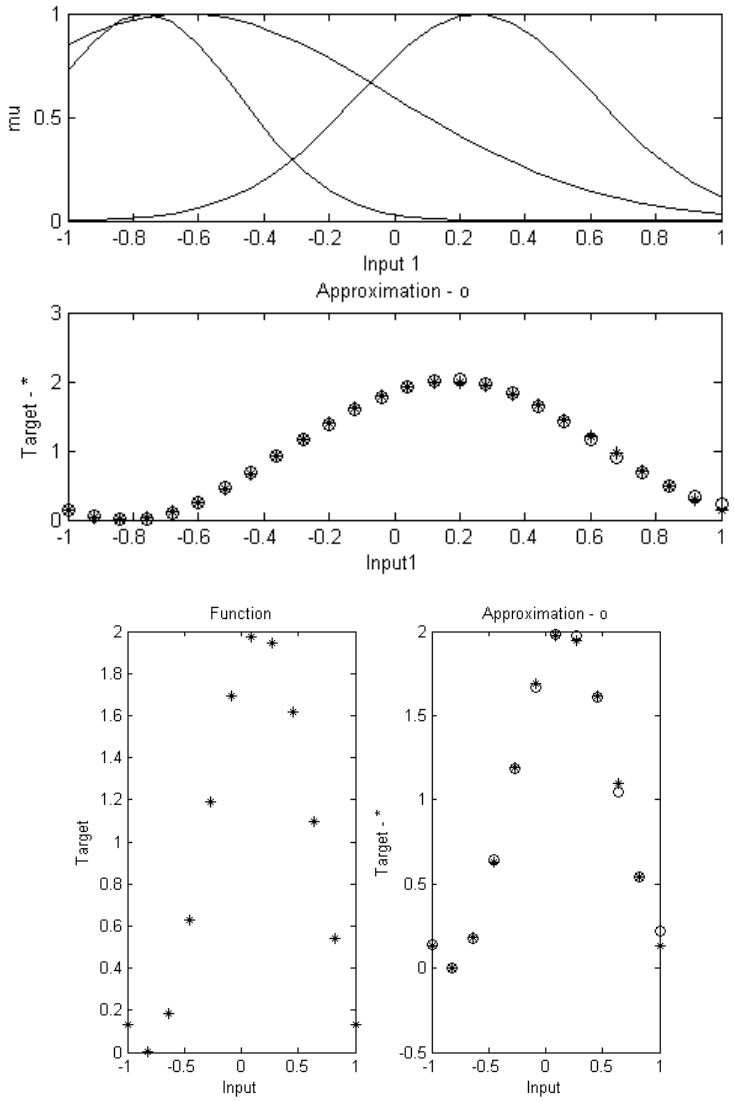
RES1 = 7.9515e-004 – на множині тестування:

p2 =-1.0000 -0.9200 -0.8400 -0.7600 -0.6800 -0.6000 -0.5200 -0.4400 -0.3600 -0.2800 -0.2000 -0.1200 -0.0400 0.0400

c = -0.6404 0.2507 -0.7659 **w** = 1.0157

sigma = 0.6298 0.3604 0.2895 1.6258

-1.0125



Теоретичні відомості

1. Нейронні мережі на основі радіальних базисних функцій Гаусса (ГРБФ-мереж**)** [5, с. 199 – 216].

Базова архітектура РБФ-мереж передбачає наявність двох шарів, де перший прихований шар виконує нелінійне перетворення вхідного простору у прихований, другий шар – лінійний. Нейрони прихованого шару реалізують набір «функцій», які формують довільний «базис» для розкладання вхідних образів (векторів). Відповідні перетворення називають радіальними базисними функціями.

Розглянемо мережу прямого поширення з одним прихованим шаром і вихідним шаром, що містить один нейрон (один нейрон у вихідному шарі

обраний спеціально для спрощення викладок без втрати спільності). Ця мережа призначена для нелінійного відображення вхідного простору в прихований, за яким слідує лінійне відображення прихованого простору в вихідний.

Найчастіше як радіальну функцію використовують функцію Гаусса. Якщо її центр розміщено в точці c_i, то вона може бути визначена як $\phi(p) = \phi(\| p - c_i \|) = \exp\left(-\frac{\| p - c_i \|^2}{2\sigma_i^2} \right)$, де c_i, p, $\sigma_i \in \Re$, σ_i – ширина функції. При цьому вихід мережі набуває вигляду (S^1 – кількість радіальних базисних функцій; ||•|| – норма, зазвичай Евкліда.)

$$y = F(p) = \sum_{i=1}^{S^1} w_i \exp\left(-\frac{\| p - c_i \|^2}{2\sigma_i^2} \right). \qquad (11.1)$$

2. Гібридний алгоритм навчання ГРБФ-мереж

У гібридному алгоритмі процес навчання поділяють на два етапи:

1) підбір лінійних параметрів мережі (ваги вихідного шару) при використанні методу псевдоінверсії;

2) адаптація нелінійних параметрів радіальних функцій (центру c_i і ширини σ_i цих функцій).

Обидва етапи тісно переплітаються. При фіксації конкретних значень центрів і ширини радіальних функцій (в перший момент це будуть початкові значення) за один крок, з використанням декомпозиції SVD, підбираються значення вектора ваги вихідного шару. Така фіксація параметрів радіальних функцій дозволяє визначити значення самих функцій $\phi_i(p_k)$ для i=1, …, S^1; k=1, …, N, де i – номер радіальної функції, а k - номер чергової навчальної пари (p_k, t_k). Чергові входи p_k генерують в прихованому шарі сигнали, які описуються векторами $\boldsymbol{\phi}(p_k) = [1, \phi_1(p_k), ..., \phi_{S^1}(p_k)]^T$. Їх супроводжує вихідний сигнал мережі $y_k = \boldsymbol{\phi}_k \mathbf{w}$, де вектор $\mathbf{w}=[w_0, w_1, ..., w_k]^T$ містить коефіцієнти ваги вихідного шару.

При наявності N пар навчання отримуємо таку систему рівнянь:

$$\begin{bmatrix} 1 & \phi_1(p_1) & \phi_2(p_1) & \dots & \phi_{S^1}(p_1) \\ 1 & \phi_1(p_2) & \phi_2(p_2) & \dots & \phi_{s^1}(p_2) \\ \dots & \dots & \dots & \dots & \dots \\ 1 & \phi_1(p_N) & \phi_2(p_N) & \dots & \phi_{s^1}(p_N) \end{bmatrix} \cdot \begin{bmatrix} w_0 \\ w_1 \\ \dots \\ w_{S^1} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_{S^1} \end{bmatrix},$$

яку можна записати у векторному вигляді як **Gw=y**.

При використанні гібридного методу на етапі підбирання ваги вихідного шару вектор **y** замінюється вектором очікуваних значень $\mathbf{t}=[t_1, t_2,\dots, t_{S1}]^T$, і утворена при цьому система рівнянь **Gw=t** вирішується за один крок з використанням псевдоінверсії $\mathbf{w}= \mathbf{G}^{+}\mathbf{t}$.

На другому етапі при зафіксованих значеннях ваги вихідного шару розраховують значення похибки для послідовності векторів p_k. Далі відбувається зворотне поширення (повернення до прихованого шару). За величиною похибки визначають вектор градієнта функції мережі щодо конкретних центрів і ширини.

Розглянемо випадок використання ГРБФ-мережі, якщо множина навчання має вигляд $\{(\mathbf{p}_i, t_i) | \ \mathbf{p}_i \in \Re^R, \ t_i \in \Re, \ i=1,2, \dots, N\}$. При цьому кожна радіальна функція визначається як $\phi_i(\mathbf{p}_k) = \exp\left(-\frac{1}{2}u_{ik}\right)$, сумарний сигнал нейрона описується виразом $u_{ik} = \sum_{j=1}^{N} \frac{(p_{jk} - c_{ij})^2}{\sigma_{ij}^2}$, а цільову функцію можна задати у вигляді $E = \frac{1}{2}\sum_{k=1}^{N}[y_k - t_k]^2 = \frac{1}{2}\sum_{k=1}^{N}\left[\sum_{i=0}^{S^1} w_i \phi_i(\mathbf{p}_k) - t_k\right]^2$.

В результаті диференціювання цієї функції отримуємо

$$\frac{\partial E}{\partial c_{ij}} = \sum_{k=1}^{N}\left[(y_k - t_k) w_i \exp\left(-\frac{1}{2}u_{ik}\right)\frac{(p_{jk} - c_{ij})}{\sigma_{ij}^2}\right];$$

$$\frac{\partial E}{\partial \sigma_{ij}} = \sum_{k=1}^{N}\left[(y_k - t_k) w_i \exp\left(-\frac{1}{2}u_{ik}\right)\frac{(p_{jk} - c_{ij})^2}{\sigma_{ij}^3}\right]$$

Застосування градієнтного методу найшвидшого спуску дозволяє визначити центри та ширину радіальних функцій за формулами: $c_{ij}(n+1) = c_{ij}(n) - \eta \frac{\partial E}{\partial c_{ij}}$, $\sigma_{ij}(n+1) = \sigma_{ij}(n) - \eta \frac{\partial E}{\partial \sigma_{ij}}$. Уточнення нелінійних параметрів радіальної функції завершує черговий цикл навчання. Багаторазове повторення обох етапів веде до навчання мережі.

На практиці виділені етапи по різному впливають на адаптацію параметрів. Зазвичай, швидше функціонує алгоритм SVD (він за один крок знаходить локальний мінімум функції). Для вирівнювання цієї диспропорції одне уточнення лінійних параметрів супроводжується декількома циклами адаптації нелінійних параметрів.

Розділ 4. Методи нечіткого моделювання (моделі із самоналаштуванням параметрів нечітких моделей)

Тема 1. Нейронечітка мережа ANFIS

12. Моделювання на основі системи Anfis пакету Fuzzy Logic Toolbox

Мета роботи: ознайомитися з принципами функціонування та навчання нейронечіткої моделі у вигляді системи Anfis (об'єднання нейронних мереж з нечіткими системами).

Об'єкт дослідження: система Anfis (anfisedit) в середовищі МатЛаб. *Предмет дослідження*: використання системи Anfis для розв'язання задач класифікації та апроксимації на основі даних вимірювань про вхід–вихід системи.

Питання для опрацювання

1. Нейронечітка мережа Anfis [2, 17].

1. Завдання до практичної роботи

1. Вивчити теоретичний матеріал.

2. Сформувати нейронечітку модель Anfis в середовищі МатЛаб, для цього потрібно:

- спроектувати апроксиматор у вигляді нечіткої моделі Сугено;
- на основі нечіткої моделі Сугено спроектувати нейронечітку модель,
- перевірити модель на здатність розв'язувати задачу апроксимації функції.

2. Контрольні запитання та завдання

1. Опишіть принцип функціонування системи Anfis.

Задачі для самостійного розв'язання: комп'ютерний практикум № 4

Аудиторна робота

Приклад 12.1. Розглянемо задачу моделювання процесу появи тріщини у склянці, яку нагріли до визначеної температури, наливши в неї воду певної температури (відомо: чим більше різниця температур, тим більше ймовірність появи тріщини).

Формуємо структуру ANFIS-мережі з п'яти шарів: 1-й шар є вхідним (L1), 5-й – вихідним (L5), з 2-го по 4-й – прихованим (L2, L3, L4)

L1: вхідний шар. Параметри ФН для нечіткої множини «холодно»: a_c=0 та b_c = 30; для нечіткої множини «жарко»: a_h = 100 та b_h = 30;

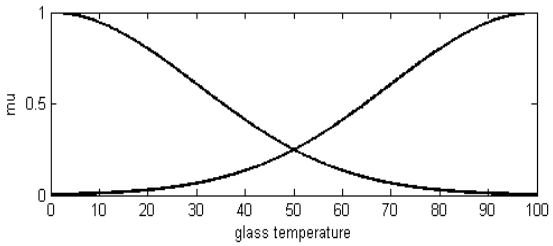
$A_1(x)=msf(a_c, b_c, x)=exp(-0.5\cdot(x-a_c)^2/b_c^2)$ – нечітка множина «холодний»;

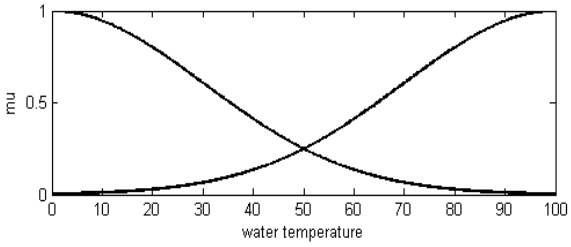
$A_2(x)=msf(a_h, b_h, x)=exp(-0.5*(x-a_h)^2/b_h^2)$ – нечітка множина «горячий».

Визначаємо значення ФН при конкретних значеннях вхідних змінних: α_i для входів із *in*:

L11=msf(a_c, b_c, in(1,*i*)); % належність стакану до нечіткої множини «холодний»;

L12=msf(a_h, b_h, in(1,*i*)); % належність стакану до нечіткої множини «горячий»;
L13=msf(a_c, b_c, in(2,*i*)); % належність води до нечіткої множини «холодна»;
L14=msf(a_h, b_h, in(2,*i*)); % належність води до нечіткої множини «гаряча».





L2: другий (прихований) шар охоплює базу правил з такими правилами:

R1: ЯКЩО «x1=холодний стакан» ТА «холодная вода», ТО y=z1
R2: ЯКЩО «x1=холодний стакан» ТА «гаряча вода», ТО y=z2
R3: ЯКЩО «x1=гарячий стакан» ТА «холодна вода», ТО y=z2
R4: ЯКЩО «x1=гарячий стакан» ТА «гаряча вода», ТО y=z3

де $z1(i1,i1)=\left(\frac{abs(i1-i2)}{100}\right)^2$; $z2(i1,i1)=\left(\frac{abs(i1-i2)}{100}\right)$; $z3(i1,i1)=1,5\cdot\left(\frac{abs(i1-i2)}{100}\right)^2$ та визначає ступінь істинності передумов кожного з цих правил на основі Т-норми.

Використовуючи оператор PROD (алгебраїчний добуток) для кожного правила, визначимо Т–норму:

L21 = L11 * L13; % холодний стакан і холодна вода
L22 = L11 * L14; % холодний стакан і гаряча вода
L23 = L12 * L13; % гарячий стакан і холодна вода
L24 = L12 * L14; % гарячий стакан і гаряча вода

L3: третій (прихований) шар для нормалізації ступенів істинності передумов кожного правила:

L31 = L21 / (L21 + L22 + L23 + L24);
L32 = L22 / (L21 + L22 + L23 + L24);
L33 = L23 / (L21 + L22 + L23 + L24);
L34 = L24 / (L21 + L22 + L23 + L24);

L4: четвертий (прихований) шар обчислює значення висновків за кожним правилом (підраховує можливість виникнення тріщини відповідно до правил R1÷ R4):

L41 = L31 * z1(in(1,i), in(2,i)); L42 = L32 * z2(in(1,i), in(2,i));
L43 = L33 * z2(in(1,i), in(2,i)); L44 = L34 * z3(in(1,i), in(2,i));

L5: **вихідний шар** визначає вихід моделі (чим більше значення L5, тим більше ймовірність виникнення тріщини):

L5 = (L41 + L42 + L43 + L44)/(L21 + L22 + L23 + L24);

Результат моделювання

Множина даних «вхід-вихід»:

[[80 18 0.7], [0 100 0.95], [25 70 0.50], [100 80 0.20], [100 0 0.95]]

Вихід: результат апроксимації на основі нечіткої моделі

$L5_1$ =0.8575; $L5_2$ =0.9942; $L5_3$ =0.8314; $L5_4$ =0.0785; $L5_5$ =0.9942

```
function  lab_rob
% обчислення ймовірності виникнення тріщини у нагрітому до визначеної
% температури стакані за наявності в ньому води визначеної температури
% (чим більше різниця - тим більше ймовірність виникнення тріщини)
%--------------------------------
clc;clear all;close all;
% параметри ФН для нечіткої множини "холодно"
ac = 0;bc = 30;
% параметри ФН для множини "гаряча"
ah = 100;bh = 30;
% вхідні дані: перший стовпчик - температура стакану,
% другий стовпчик - температура води
in = [[80,  0,   25,  100, 100];
  [18, 100,  70, 80,  0] ;
  [0.7, 0.95, 0.5, 0.2, 0.95 ]]; % Величини, необхідні для
% налаштування параметрів ФН під час навчання нейронечіткої мережі
function y=z1(a,b)
 y=(abs(a-b)/100)^2;
end
function y=z2(a,b)
 y=(abs(a-b)/100);
end
function y=z3(a,b)
 y=1.5*(abs(a-b)/100)^2;
end
function y=msf(a,b,u)
% розраховує значення ФН із заданими параметрами
 y = exp(-0.5*((u-a)/b).^2);
end

x = 0:0.2:100; % універсум
a1=0;b1=30; y1 = exp(-0.5*((x(:)-a1)/b1).^2);
a2=100;b2=30; y2=exp(-0.5*((x(:)-a2)/b2).^2);
figure(1);subplot(2,1,1)
Attempt=plot(x(:),y1(:),'-k','linewidth',2); hold on
Attempt=plot(x(:),y2(:),'-k','linewidth',2); hold off
ylabel('mu');  xlabel('glass temperature');
subplot(2,1,2)
Attempt=plot(x(:),y1(:),'-k','linewidth',2); hold on
Attempt=plot(x(:),y2(:),'-k','linewidth',2); hold off
ylabel('mu');  xlabel('water temperature');
% Формуємо структуру нейромережі із 5 шарів: 1й - вхідний (L1), 5й
- вихідний (L5),
% 2й-4й - прихований шар (L2,L3,L4)
```

```
for (i=1:length(in(1,:)))
% Вхід (L1)
% L11, L12 - 1-й,2-й нечіткі нейрони відповідають за прийняття
сигналу про t стакану
L11 = msf(ac, bc, in(1,i)); % належність стакану до НМ "холодний"
L12 = msf(ah, bh, in(1,i)); % належність стакану до НМ "гарячий"
% L13,L14 - 2й,4й нечеткі нейрони відповідають за прийняття
сигналу про t води
L13 = msf(ac, bc, in(2,i)); % належність води до НМ "холодний"
L14 = msf(ah, bh, in(2,i)); % належність води до НМ "гарячий"
% Другий шар (L2) охоплює базу правил с 4 правилами:
% R1:холодний стакан і холодна вода
% R2:холодний стакан і гаряча вода
% R3:гарячий стакан і холодна вода
% R4:гарячий стакан і гаряча вода
%---------------------------------
% Обчислюємо Т-норму для чого використаємо оператор PROD
% Отримано нейрони L21-L24 - нейрони правил
L21 = L11 * L13; % холодний стакан і холодна вода
L22 = L11 * L14; % холодний стакан і гаряча вода
L23 = L12 * L13; % гарячий стакан і холодна вода
L24 = L12 * L14; % гарячий стакан і гаряча вода
% Прихований шар (L3) для нормування правил
L31 = L21 / (L21 + L22 + L23 + L24);
L32 = L22 / (L21 + L22 + L23 + L24);
L33 = L23 / (L21 + L22 + L23 + L24);
L34 = L24 / (L21 + L22 + L23 + L24);
% Прихований шар (L4) для підрахунку можливості виникнення
% тріщины
L41 = L31 * z1(in(1,i),in(2,i));
L42 = L32 * z2(in(1,i),in(2,i));
L43 = L33 * z2(in(1,i),in(2,i));
L44 = L34 * z3(in(1,i),in(2,i));
% Визначаємо відповідь: чим більше L5, тим більше ймовірність
% виникнення тріщини
L5 = (L41 + L42 + L43 + L44)/(L21 + L22 + L23 + L24)
end; end
```

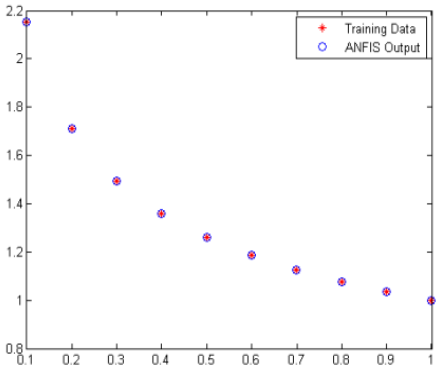
Приклад 12.2. Апроксимація функції $y = 1/\sqrt[3]{x},\ x \in [0.1\ 1]$.

```
clear all; clc; close all;
X=[];Xtest=[]; dx=0.1;n=10; X(1)=dx; Xtest(1)=dx*(3/2);
for i=2:n
 X(i)=X(i-1)+dx; Xtest(i)=X(i)+dx/2;
end
X, Xtest
Y=1./X.^(1/3); Ytest=1./Xtest.^(1/3);
data=[X;Y]; data=data';
testData=[Xtest(1:end-1);Ytest(1:end-1)]';
figure (1); plot(data(:,1),data(:,2), 'linewidth',2)
title('y=1/x\^(1/3);'); xlabel('X');ylabel('Y');
inFIS = genfis1(testData,3,'trimf');
```

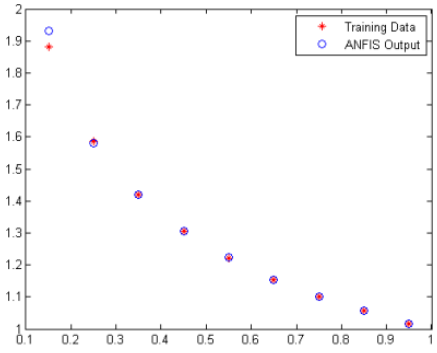
```
fis = anfis(data,inFIS,40); writefis(fis)

figure (2);anfisOutput = evalfis(data(:,1),fis);
plot(data(:,1),data(:,2),'*r',data(:,1),anfisOutput,'ob',
'linewidth',2)
legend('Training Data','ANFIS Output','Location','NorthEast')
[n,m]=size(anfisOutput);
err1=sum(abs(anfisOutput-data(:,2)))/n

figure (3); anfisOutput = evalfis(testData(:,1),fis);
plot(testData(:,1),testData(:,2),'*r',testData(:,1),anfisOutput,'o
b', 'linewidth',2)
legend('Training Data','ANFIS Output','Location','NorthEast')
[n,m]=size(anfisOutput);
err2=sum(abs(anfisOutput-testData(:,2)))/n
```

Результат моделювання

X = [0.1000 0.2000 0.3000 0.4000 0.5000 0.6000 0.7000 0.8000 0.9000 1.0000]
Xtest =[0.1500 0.2500 0.3500 0.4500 0.5500 0.6500 0.7500 0.8500 0.9500 1.0500]



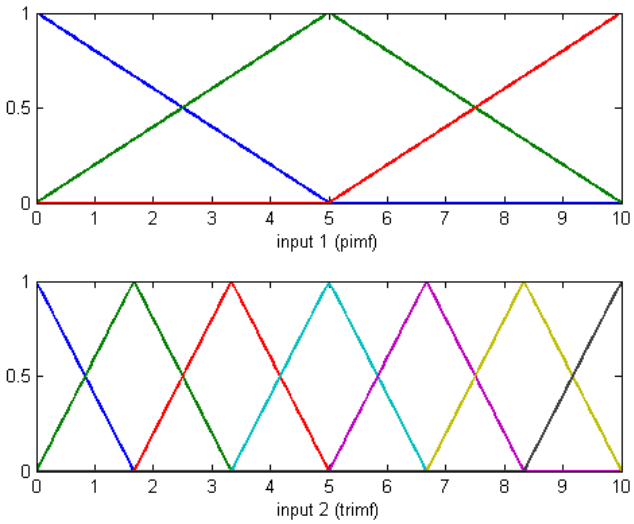
Моделювання на множині навчання



Моделювання на множині тестування

Приклад 12.3. Приклад використання функції *anfis* із двома аргументами

```
x1 = (0:0.1:10)'; x2 = (0:0.1:10)';
y = (sin(2*x1)+x2)./exp(x2/5); trnData=[x1 x2 y];
fis = genfis1(trnData,[3 7],char('trimf','trimf'));
[x,mf] = plotmf(fis,'input',1);
subplot(2,1,1), plot(x,mf, 'linewidth',2);
xlabel('input 1 (pimf)');
[x,mf] = plotmf(fis,'input',2);
subplot(2,1,2), plot(x,mf, 'linewidth',2);
xlabel('input 2 (trimf)');
writefis (fis,'ll_5'); out_fis = anfis(trnData,fis,20);
```



Приклад 12.4. Застосування функції *anfis* для розв'язання задачі апроксимації із використанням нечіткої моделі Сугено

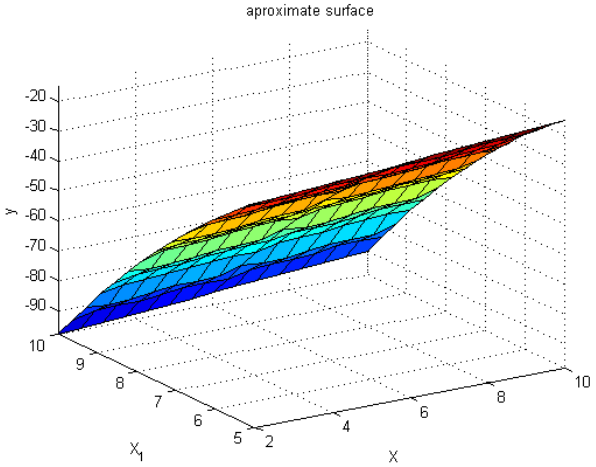
```
script %function [X,X_1,Y,out, ep]=lab_2var
clc;clear all;close all;
% початкові дані
f_name='l_12';
x_t=5; x_beg=2; x_end=10;
x_t_1=6; x_beg_1=5; x_end_1=10;
n=x_t;  m=x_t_1; %кількість термів
nm=n*m;
% ----Формування моделі Сугено-------
% формування вхідних чітких векторів X і X_1
X=x_beg:(x_end-x_beg)/(n -1):x_end;
delta_x=(x_end-x_beg)/(n-1)
X_1=x_beg_1:(x_end_1-x_beg_1)/(m-1):x_end_1;
delta_x1=(x_end_1-x_beg_1)/(m-1)
% Формування масиву Y (вихідних даних)
z=0;
for i=1:n
 for j=1:m
    z=z+1; Y(z)=(X(i)-(X_1(j))^2);
 end; end
k=1;data=zeros(nm,3)
for i=1:n
  for j=1:m
   data(k,1)=X(i);data(k,2)=X_1(j)
   data(k,3)=Y(k); k=k+1;
end;end; % data

Y2=sort(Y);
% Визначення області значень вихідної змінної 'Y'
ymin=min(Y);ymax=max(Y);delta_y=(ymax-ymin)/nm;
% ---- Створення СНВ у вигляді  FIS-структури ------
ai=newfis(f_name,'sugeno','min','max','prod','sum','wtaver');
```

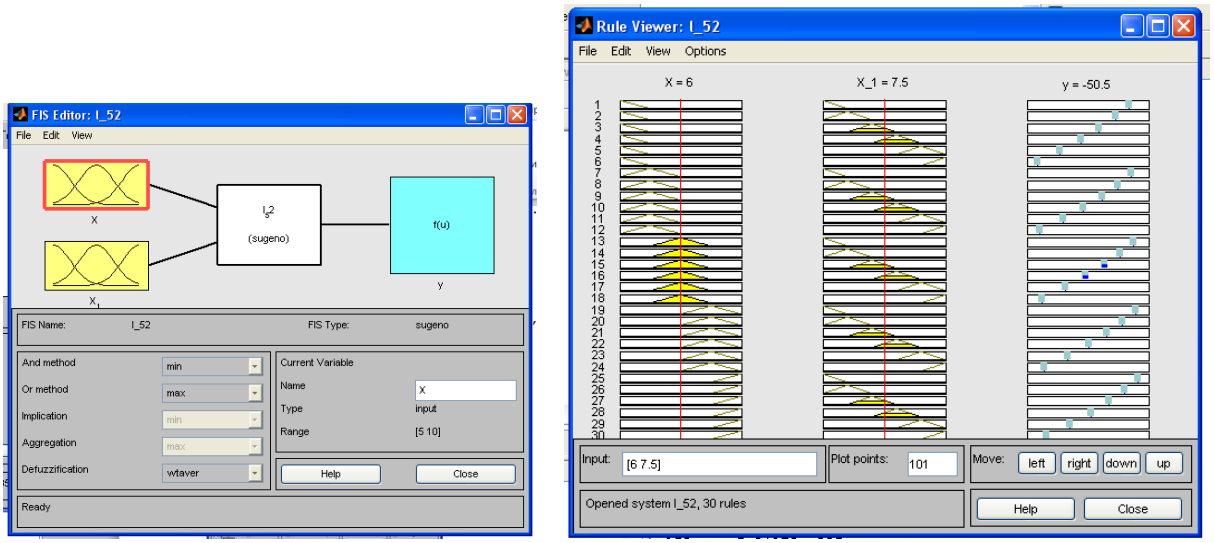
```
% Визначення термів вхідних змінних X та X_1
ai= addvar (ai,'input','X', [x_beg x_end]);
for i=1:n
 if i==n
  c=x_end+delta_x;
 else
  c=X(i+1)
 end;
 if i==1
  a=x_beg -delta_x;
 else
     a=X(i-1)
 end; b=X(i);
 ai=addmf(ai,'input',1, int2str(i),'trimf', [a b c])
end;
ai = addvar(ai,'input','X_1', [x_beg_1 x_end_1])
for i=1:m
 if i==m
  c=x_end_1 + delta_x1;
 else
   c=X_1(i+1)
 end;
 if i==1
  a=x_beg_1-delta_x1;
 else
     a=X_1(i-1)
 end;  b=X_1(i);
 ai=addmf (ai,'input',2, int2str(i),'trimf', [a  b c])
end;
% Визначення термів вихідної змінної 'Y'
ai= addvar (ai,'output','y', [ymin ymax]);
for i=1:nm
 if i==nm
  c=ymax+delta_y;
 else
     c=Y2(i+1);end;
 if i==1
  a=ymin-delta_y;
 else
     a=Y2(i-1)
 end;
 b=Y2(i);
 ai= addmf (ai,'output',1, ['y', int2str(i)],'constant', b);
end;
% Формування масиву правил на основані спостережень за графік.
 rullist=ones(nm,5); z=0;
 for i=1:n
  for j=1:m
    z=z+1; rullist(z,1)=i; rullist(z,2)=j;
  for k=1:nm
   if (Y((i-1)*m+j)==Y2(k))
    break;
```

```
  end; end;
 rullist(z,3)=k;
end; end;
ai= addrule(ai, rullist);

figure (2), gensurf (ai);title ('aproximate surface')
writefis (ai,f_name)
% Параметри нечіткого виведення
X_check=5:0.4:7; X_1_check=6;rr=length(X_check);
Y_check(1:rr)=(X_check(1:rr)-X_1_check^2)
%обчислення нев'язки моделювання функції СНВ
ai=readfis(f_name); ep=0; % showfis(ai);
for i = 1:rr
 out(i)=evalfis([X_check(i) X_1_check], ai);
 ep=ep+abs(out(i)-Y_check(i));
end;
ep=ep/rr;
% ----Використання Anfis-----
out_fismat=anfis(data, ai, 250);
% Обчислення нев'язки моделювання функції СНВ
 ai=readfis(f_name); %  showfis(ai);
 data_ch=zeros(rr,3); ep1=0;
 for i = 1:rr
   data_ch(i,1)=X_check(i); data_ch(i,2)=X_1_check;
   out(i)=evalfis([data_ch(i,1) data_ch(i,2)], out_fismat);
   ep1=ep1+abs(out(i)-Y_check(i));
   end;
ep, ep1=ep1/rr
```

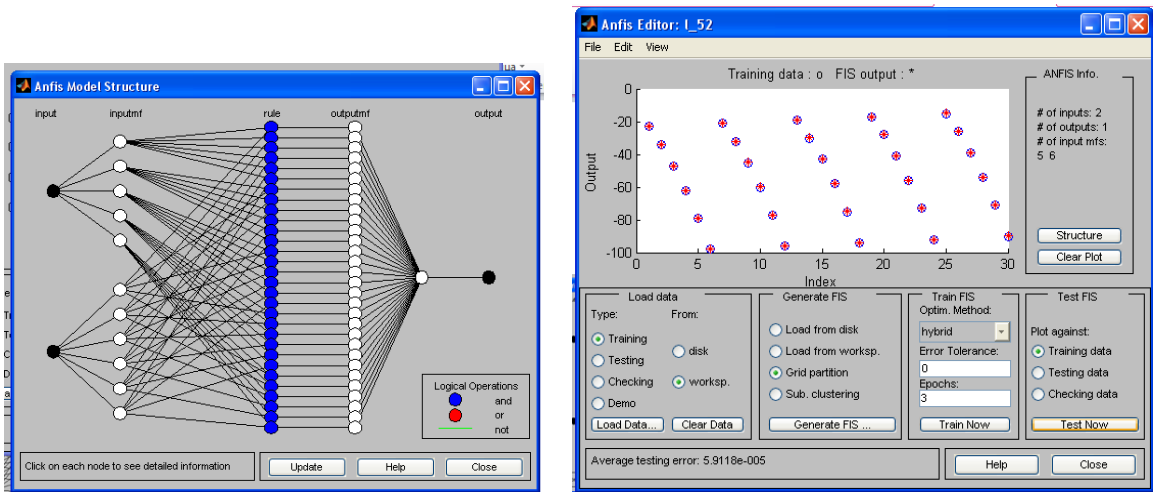
Результат моделювання

Похибка моделювання:
ep= 1.1842e-015 – на основі нечіткої моделі Сугено;
ep1= 0.0067 – на основі нейро-нечіткої моделі Anfis





Меню «EDIT» → підменю «ANFIS»



«Load Data» → data; «Test Now»

У найпізніших версіях MatLab застосовують такі команди:

```
…
ai=sugfis; % --Створення Sugeno FIS --
ai= addInput(ai, [x_beg x_end], 'Name','X'); % змінна x
ai= addInput(ai,[x_beg_1 x_end_1], 'Name','X_1'); % змінна x_1
ai= addOutput(ai, [ymin ymax], 'Name','y'); % змінна y
…
ai=addMf(ai,'X','trimf', [a b c]); …
ai=addMf (ai,'X_1','trimf', [a b c]); …
ai= addMf (ai,'y', 'constant', b); …
ai= addrule(ai, rullist); …
figure, gensurf (ai);
% Параметри нечіткого виведення
X_check=5:0.4:7; X_1_check=6;rr=length(X_check);
Y_check(1:rr)=(X_check(1:rr)-X_1_check^2)
% обчислення нев'язки моделювання функції СНВ
ep=0;
for i = 1:rr
 out(i)=evalfis([X_check(i) X_1_check], ai);
```

```
 ep=ep+abs(out(i)-Y_check(i));
end; ep=ep/rr;
```

Приклад 12.5. Застосування функції anfis для розв'язання задачі апроксимації без використання нечіткої моделі Сугено

```
clc; clear all; close all;
a_t=7; a_beg=-5; a_end=5;
b_t=6; b_beg=-1; b_end=5;
delta_a=(a_end-a_beg)/(a_t-1); A=a_beg:delta_a:a_end;
delta_b=(b_end-b_beg)/(b_t-1); B=b_beg:delta_b:b_end;
A=A'; B=B'; arr=zeros(a_t*b_t,2);z=1;
for i=1:a_t
  for j=1:b_t
     arr(z,1)=A(i); arr(z,2)=B(j);
     Y(z)=A(i)*sin(B(j)); z=z+1;
end;end;
trnData=[arr(:,1) arr(:,2) Y(:)];
fis=genfis1(trnData, [a_t b_t], char('gaussmf','gaussmf'));
out_fis=anfis(trnData,fis,100);
figure(1)%-------------
[x, mf]=plotmf(fis, 'input', 1);
subplot(2,1,1), plot(x, mf, 'linewidth',2);
title('Initial membership functions');
xlabel('input 1 (gaussmf)');
[x, mf]=plotmf(fis, 'input', 2);
subplot(2,1,2), plot(x, mf,'linewidth',2);
xlabel('input 2 (gaussmf)');
figure(2) %------------
[x, mf]=plotmf(out_fis, 'input', 1);
subplot(2,1,1), plot(x, mf, 'linewidth',2);
title('Resulting membership functions');
xlabel('input 1 (gaussmf)');
[x, mf]=plotmf(out_fis, 'input', 2);
subplot(2,1,2), plot(x, mf, 'linewidth',2);
xlabel('input 2 (gaussmf)');
A_ch=a_beg+delta_a/2:delta_a:a_end-delta_a/2;
B_ch=b_beg+delta_b/2:delta_b:b_end-delta_b/2;
rr=length(A_ch)*length(B_ch); arr1=zeros(rr,2);
z=1;Yt_ch=zeros(a_t-1, b_t-1);
for i=1:a_t-1
     for j=1:b_t-1
        Y_ch(i,j)=A_ch(i)*sin(B_ch(j));
        arr1(z,1)=A_ch(i); arr1(z,2)=B_ch(j);
        arr1(z,3)= Y_ch(i,j); z=z+1;
end;end; ep1=0;
for i = 1:rr
   out(i)=evalfis([arr1(i,1) arr1(i,2)], out_fis);
   ep1=ep1+abs(out(i)-arr1(i,3));
end; ep1=sum(ep1(:));
abs_er=ep1/((a_t-1)*(b_t-1)); disp(abs_er); z=1;
```

```
for i=1:a_t-1
  for j=1:b_t-1
    Yt_s(i,j)=out(z);z=z+1;
end;end;
figure(3); surf(B_ch,A_ch,Yt_s);title('Approximation')
figure(4); surf(B_ch,A_ch,Y_ch); title('y=a*sin(b)')
```

Результат:

`abs_er= 0.2570` – похибка моделювання у точках [A_ch B_ch]

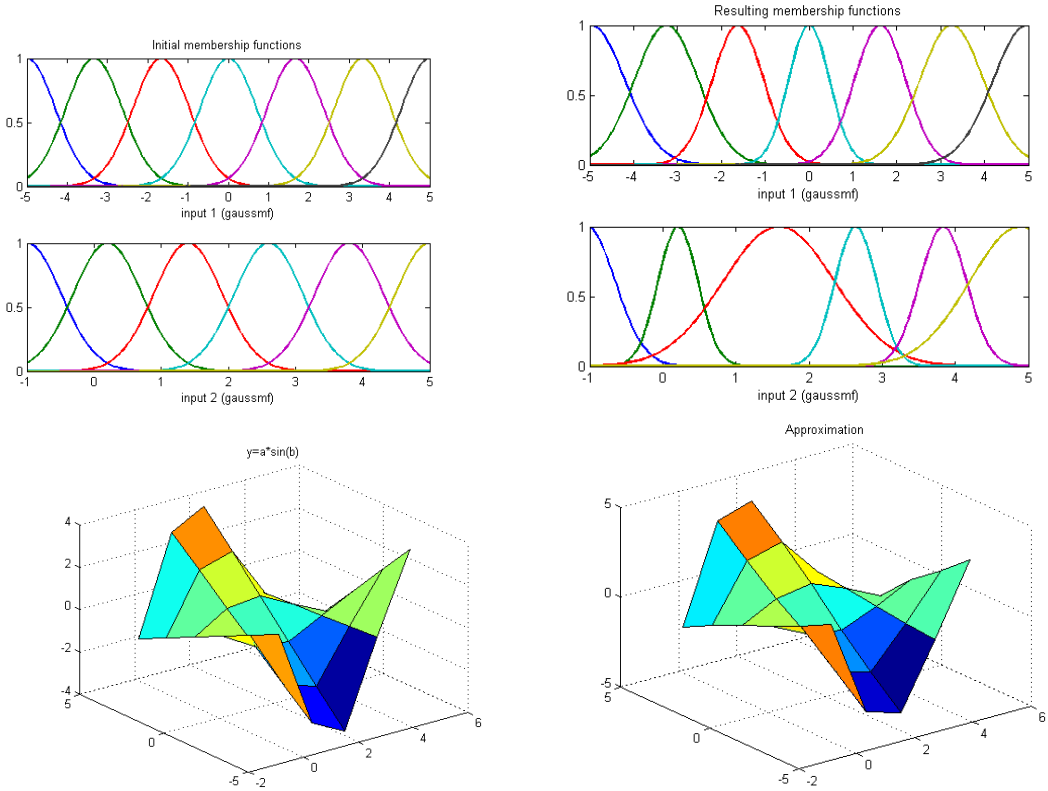


Рис. 12.1. Графіки реальної функції та результату апроксимації

Теоретичний матеріал

1. Нейронечітка мережа Anfis [2, с. 97–105; 17]. Покажемо, як побудувати гібридну нейронну мережу (НМ), адаптивну мережу за Jang, яка функціонально еквівалентна механізму логічного виведення Сугено. Сугено і Такагі використали такі правила:

R1: if x is A1 and y is B1 then $z1=a1x+b1y$

R2: if x is A2 and y is B2 then $z2=a2x+b2y$

Рівні запуску цих правил обчислюють як:

α_1 = A1($x0$) × B1($y0$), α_2 = A2 ($x0$) × B2($y0$),

де логічну функцію «and» можна змоделювати будь-якою неперервною Т-нормою: α_1 = A1($x0$) $\wedge$ B1($y0$), α_2 = A2 (x0) $\wedge$ B2 (y0). Тоді окремі виходи правила випливають із співвідношень: $z1 = a1x0 + b1y0$, $z2 = a2x0 + b2y0$, а чіткий керуючий вплив має вигляд $z_0 = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1 + \alpha_2} = \beta_1 z_1 + \beta_2 z_2$, де β1 і β2 – нормовані значення α1 і α2 по відношенню до суми (α_1+α_2), тобто $\beta_1 = \frac{\alpha_1}{\alpha_1 + \alpha_2}$, $\beta_2 = \frac{\alpha_2}{\alpha_1 + \alpha_2}$. Гібридну НМ, яка обчислювально ідентична цьому типу міркувань, зображено на рис. 12.3.

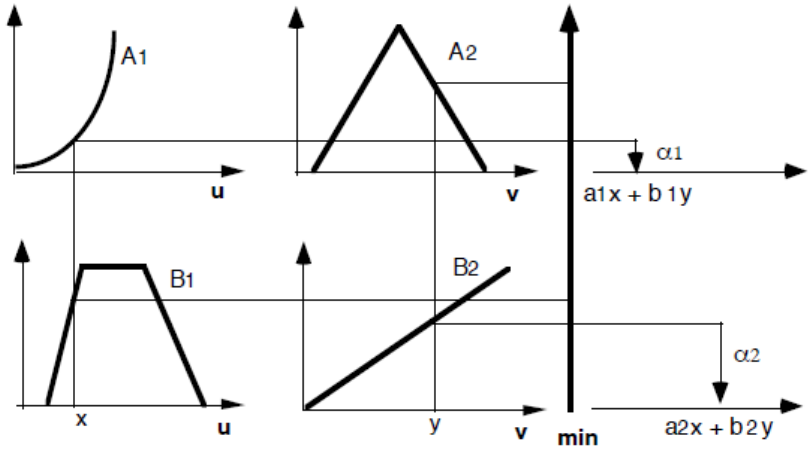


Рис. 12.2. Механізм логічного виведення Сугено

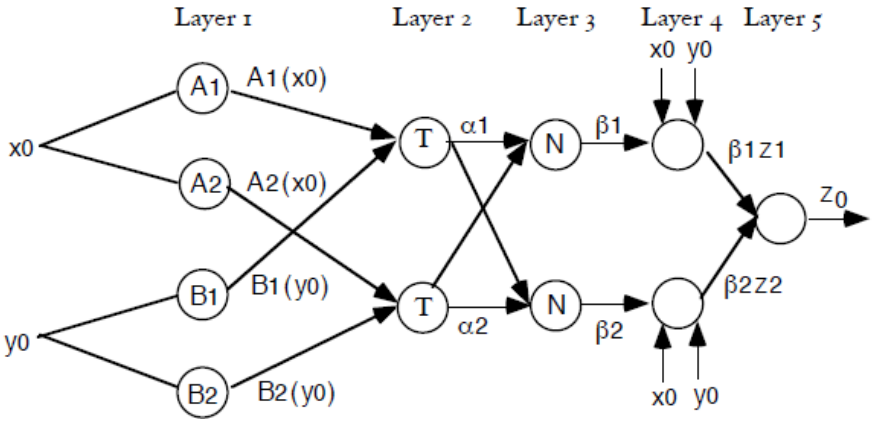


Рис. 12.3. ANFIS архитектура методу суджень Сугено

Припустимо наявність тільки двох правил і двох лінгвістичних значень для кожної вхідної змінної.

Шар 1. Вихід вузла – це ступінь, з якою даний вхід задовольняє лінгвістичної мітці, пов'язаній з цим вузлом. Як правило, для зображення лінгвістичних термів вибирають ФН у вигляді функції Гаусса: $A_i(u) = \exp\left[-\frac{1}{2}\left(\frac{u-a_{i1}}{b_{i1}}\right)^2\right]$, $B_i(u) = \exp\left[-\frac{1}{2}\left(\frac{u-a_{i2}}{b_{i2}}\right)^2\right]$, де $\{a_{i1}, a_{i2}, b_{i1}, bi2\}$ — набір параметрів (будь-яка неперервна, наприклад, трапецієподібна і трикутна форми ФН, також є кандидатами для функцій вузлів у цьому шарі). Оскільки значення цих параметрів змінюються, то функція належності змінюється відповідно, демонструючи тим самим різні форми на лінгвістичних змінних A_i і B_i. Параметри в цьому шарі, називаються параметрами передумови.

Шар 2. Кожен вузол обчислює силу запуску відповідного правила. Вихідний сигнал верхнього нейрона має вигляд:

$$\alpha_1 = A1\,(x0) \times B1\,(y0) = A1(x0) \wedge B1(y0),$$

а вихід нижнього нейрона – $\alpha_2 = A2(x0) \times B2(y0) = A2(x0) \wedge B2(y0)$.

Обидва вузли в цьому шарі, які називають вузлами правил, позначені символом «Т».

Шар 3. Кожен вузол в цьому шарі позначений «N», щоб вказати нормалізацію рівнів запуску. Вихідний сигнал верхнього нейрона – це нормований (по відношенню до суми рівнів запуску) рівень запуску першого правила $\beta_1 = \frac{\alpha_1}{\alpha_1 + \alpha_2}$, а вихід нижнього нейрона – нормований рівень запуску другого правила $\beta_2 = \frac{\alpha_2}{\alpha_1 + \alpha_2}$.

Шар 4. Вихід верхнього нейрона є нормалізованим добутком рівня запуску і виходу першого правила вигляду $\beta_1 z_1 = \beta_1(a_1 x0 + b_1 y0)$. Вихідний сигнал нижнього нейрона є нормалізованим добутком рівня запуску і індивідуального виходу другого правила вигляду $\beta_2 z_2 = \beta_2\,(a_2 x0 + b_2 y0)$.

Шар 5. Один вузол в цьому шарі обчислює загальний вихід системи як суму всіх вихідних сигналів, тобто $z0 = \beta_1 z_1 + \beta_2 z_2$.

13–14. Застосування нейронечіткої моделі ANFIS на основі алгоритмів Ванга-Менделя і Такагі–Сугено–Канга

Мета роботи: ознайомитися з принципами функціонування та навчання нейронечіткої моделі Anfis на основі алгоритмів Ванга–Менделя, Такагі–Сугено–Канга. *Об'єкт дослідження*: нечіткі НМ Ванга–Менделя, Такагі–Сугено–Канга.

Питання для опрацювання

1. Нечітка нейронна мережа Ванга-Менделя [2].

1. Завдання до практичної роботи

1. Вивчити теоретичний матеріал.
2. Розробити нейромережу ANFIS на основі методу Ванга–Менделя.

2. Контрольні запитання та завдання

1. Опишіть структуру та алгоритм навчання нейронечіткої моделі Anfis на основі алгоритму Ванга–Менделя.
2. Опишіть структуру та алгоритм навчання нейронечіткої моделі Anfis на основі алгоритму Такагі–Сугено–Канга [2, с. 113–119]

Задачі для самостійного розв'язання: *комп'ютерний практикум* № 7

Аудиторна робота

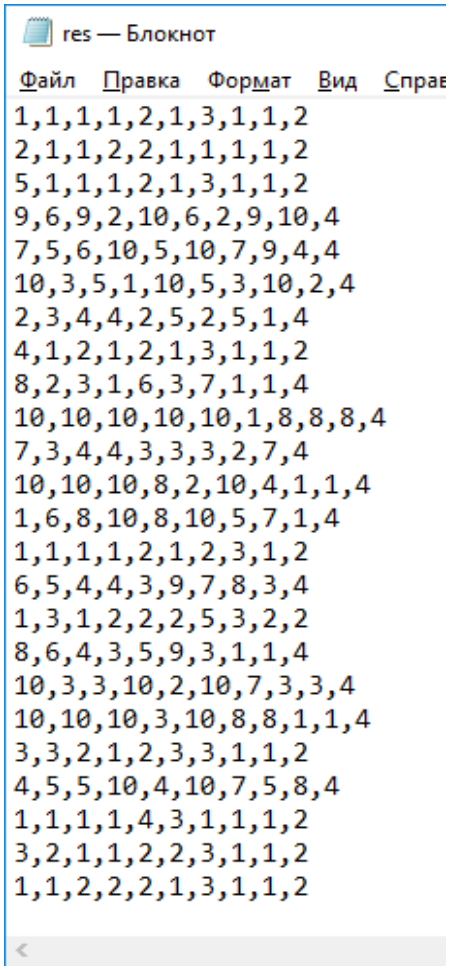
Приклад 13.1: *алгоритм Ванга-Менделя*. Програма реалізації нейронечіткої мережі для діагностики раку молочної залози, розроблена мовою програмування Python.

```
import math
import matplotlib.pyplot as plt

def gaussmf(x,a,b):
    return math.exp(-(x-a)**2/b**2)
#Правила виведення
prin=[0,0,0,0,0,0,0,0,0]; H=[[0,0,0,0,0,0,0,0,0]]
while 0 in prin:
    for j in range(9):
        if prin[8-j]==0:
            prin[8-j]=1;break
        else:
            prin[8-j]=0
    H.append(prin.copy())
with open("D:\\res.txt") as f:
    data=[i.strip() for i in f.readlines()]
    data=[i.split(",") for i in data]
    data=[[int(j) for j in i] for i in data]
xtech=[i.copy() for i in data[:400]]
#Ініціалізація
```

```
    n=512
    xt=[2 for i in range(9)]
    xbeg=[1 for i in range(9)]; xend=[10 for i in range(9)]
    xdelta=[(xend[i]-xbeg[i])/(xt[i]-1) for i in range(9)]
    A=[[xbeg[j]+H[i][j]*xdelta[j] for j in range(9)] for i in
range(n)]
    B=[[xdelta[j]/2 for j in range(9)] for i in range(n)]
    C=[1/n for i in range(n)]
    eps=0.1;nu=0.5
    delta=1; itter=0;K=len(xtech)
    #Навчання
    print("iter start")
    while delta>eps and itter<500:
        print(itter, delta)
        delta=0
        for k in range(K):
            mu=[1 for i in range(n)]
        for i in range(n):
            for j in range(9):
                mu[i]=mu[i]*gaussmf(xtech[k][j],A[i][j],B[i][j])
            y=sum(C[i]*mu[i] for i in range(n))/sum(mu)
            dels=abs(y-xtech[k][9])
            if dels>delta:
                delta=dels
            if delta>eps:
                for i in range(n):
                    C[i]=C[i]-nu*(y-xtech[k][9])*mu[i]/sum(mu)
        for k in range(K):
            mu=[1 for i in range(n)]
        for i in range(n):
            for j in range(9):
                mu[i]=mu[i]*gaussmf(xtech[k][j],A[i][j],B[i][j])
            y=sum(C[i]*mu[i] for i in range(n))/sum(mu)
        for i in range(n):
            for j in range(9):
                A[i][j]=A[i][j]-nu*2*(xtech[k][j]-A[i][j])*(y-
xtech[k][9])*(C[i]-y)*mu[i]/(B[i][j]**2*sum(mu))
                B[i][j]=B[i][j]-nu*2*(xtech[k][j]-A[i][j])**2*(y-
xtech[k][9])*(C[i]-y)*mu[i]/(B[i][j]**2*sum(mu))
        itter=itter+1;
    print("iter_end")
    #Тестування
    xtest=data[0:200];K=len(xtest)
    print("---")
    K2true,K2false,K4true,K4false=0,0,0,0
    for k in range(K):
        mu=[1 for i in range(n)]
        for i in range(n):
            for j in range(9):
                mu[i]=mu[i]*gaussmf(xtest[k][j],A[i][j],B[i][j])
        y=sum(C[i]*mu[i] for i in range(n))/sum(mu)
        if y<1:
```

```
            y=2
        else:
            y=4
        if y==xtest[k][9]:
            if y==2:
                K2true+=1
            else:
                K4true+=1
        else:
            if y==2:
                K2false+=1
            else:
                K4false+=1
print(K2true,K2false,K4true,K4false);input()
```

Файл із вхідними даними:



Результат роботи програми:

Програма завершила роботу на 245 ітерації навчання

```
240 0.10110710549214286
241 0.10089234522920631
242 0.10067892021177904
243 0.10046681668161517
244 0.10025602107829124
245 0.10004652003560932
iter_end
---
17 1 19 3
```

Ln: 255 Col: 0

З 40 вхідних об’єктів 18 було класифіковано, як елементи першої з кодом «2» групи (17 правильно, 1 хибно) та 22 об’єкти, як елементи другої групи з кодом «4» (19 правильно, 3 неправильно).

Для кожної із дев’яти вхідних змінних було побудовано дві функції належності.

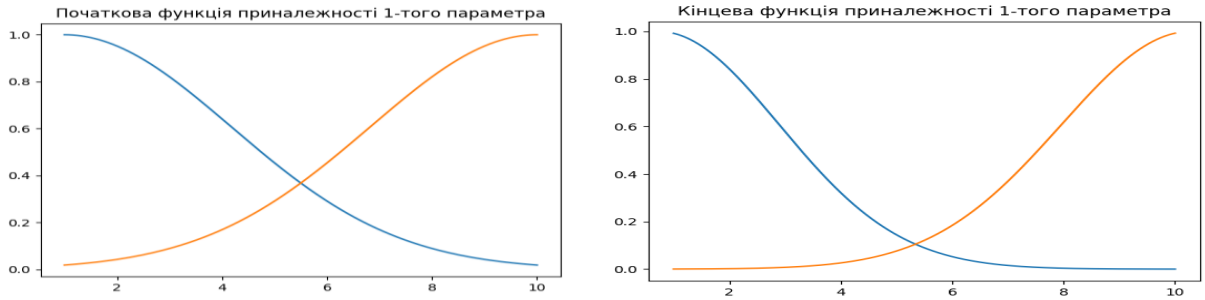


Рис. 13.1. Початковий і кінцевий графіки ФН першого параметру

Теоретичні відомості

1. Нечітка нейронна мережа Ванга-Менделя [2, с. 105–113]. Ця мережа реалізує нечітку модель, засновану на правилах вигляду:

$$R_i: \text{ЯКЩО } x_1 \in A_{il} \text{ I } \ldots \text{ I } x_m \in A_{im}, \text{ ТО } y \in B_i;\ i=1, \ldots, n.$$

Алгоритм нечіткого виведення, який виконується цією НМ, базується на таких положеннях: 1) вхідні змінні зміні є чіткими; 2) ФН всіх нечітких множин подано функцією Гауса; 3) нечітка імплікація – нечіткий добуток; 4) Т-норма – нечіткий добуток; 5) акумулювання активізованих висновків правил не проводиться; 6) метод дефазифікації – середній центр. Таким чином, нечітку модель та механізм нечіткого виведення для цієї нечіткої нейронної мережі можна відобразити таким виразом:

$$\mu_{B'_i}(y) = \sup_{x \in X} \left\{ \mu_{A'_i}(x) \mathrm{T} \mu_{A_i \to B_i}(x, y) \right\} =$$

$$= \sup_{x \in X} \left\{ \mu_{A'_i}(x) \cdot \mu_{A_i \to B_i}(x, y) \right\} = \sup_{x \in X} \left\{ \mu_{A'_i}(x) \cdot \mu_{A_i}(x) \cdot \mu_{B_i}(y) \right\} =$$

$$= \sup_{x_1, \ldots, x_m \in X} \left\{ \mu_{B_i}(y) \prod_{j=1}^{m} \mu_{A'_{ij}}(x_j) \mu_{A_{ij}}(x_j) \right\} \qquad (13.1)$$

В даному випадку вхідні змінні x_1, …, x_m є чіткими, тому вираз (13.1) має такий вигляд:

$$\mu_{B'_i}(y) = \mu_{B_i}(y) \prod_{j=1}^{m} \mu_{A_{ij}}(x'_j) \qquad (13.2)$$

Враховуючи, що акумулювання активованих висновків правил не проводиться, а методом дефазифікації є метод середнього центра, то дефазифіковане значення вихідної змінної визначається за формулою:

$$y' = \sum_{i=1}^{n} \left(\arg \max_y \mu_{B'_i}(y) \cdot \mu_{B'_i}(y) \right) =$$

$$= \frac{\sum_{i=1}^{n} \left(\arg \max_y \mu_{B'_i}(y) \cdot \mu_{B_i}(y) \cdot \prod_{j=1}^{m} \mu_{A_{ij}}(x'_j) \right)}{\sum_{i=1}^{n} \left(\mu_{B_i}(y) \cdot \prod_{j=1}^{m} \mu_{A_{ij}}(x'_j) \right)} \qquad (13.3)$$

Максимальне значення, яке $\mu_{Bi}(y)$ може прийняти у точці $arg \max_y \mu_{B'_i}(y)$, дорівнює «1», тому вираз (13.3) можна переписати:

$$y' = \frac{\sum_{i=1}^{n} \left(\arg \max_y \left(\mu_{B'_i}(y) \right) \cdot \prod_{j=1}^{m} \mu_{A_{ij}}(x'_j) \right)}{\sum_{i=1}^{n} \prod_{j=1}^{m} \mu_{A_{ij}}(x'_j)} \qquad (13.4)$$

У нашому випадку функції належності всіх нечітких множин подані функцією Гауса, тому вираз (13.4) має вигляд:

$$y' = \frac{\sum_{i=1}^{n} \left(\arg \max_y \left(\exp\left(-\frac{y - c_i}{d_i} \right) \right) \cdot \prod_{j=1}^{m} \exp\left(-\frac{x'_j - a_{ij}}{b_{ij}} \right) \right)}{\sum_{i=1}^{n} \prod_{j=1}^{m} \exp\left(-\frac{x'_j - a_{ij}}{b_{ij}} \right)} \qquad (13.5)$$

де c_i, d_i – відповідно центри і ширина гаусових функцій, які зображують ФН нечітких множин B_i висновків правил: a_{ij}, b_{ij} – відповідно центри і ширина гаусових функцій, які зображують ФН нечітких множин A_i передумов правил. У кінцевому вигляді маємо:

$$y' = \frac{\sum_{i=1}^{n}\left(c_i \prod_{j=1}^{m} \exp\left(-\left(\frac{x'_j - a_{ij}}{b_{ij}}\right)^2\right)\right)}{\sum_{i=1}^{n}\prod_{j=1}^{m} \exp\left(-\left(\frac{x'_j - a_{ij}}{b_{ij}}\right)^2\right)}. \qquad (13.6)$$

Структура даної нечіткої мережі складається з чотирьох шарів (рис. 13.1, де $\mu_{Al}(x_1)=(\mu_A^{(l)}(x_1), \ldots, \mu_{Al}(x_m) = (\mu_A^{(M)}(x_m))$.

Шар 1 виконує фазифікацію вхідних змінних x'_j (j=1, …, m). Елементи цього шару обчислюють значення ФН $\mu_{Aij}(x'_j)$, заданих гаусовими функціями з параметрами a_{ij} і b_{ij}.

У *шарі* 2, кількість елементів якого дорівнює кількості правил в базі правил n (n=M), виконується агрегування ступенів істинності передумов відповідних правил.

У *шарі* 3 перший елемент служить для активації висновків правил (c_i) у відповідності зі значеннями агрегованих на попередньому шарі ступенів істинності передумов правил.

Шар 4, який складається з одного елемента, виконує дефазифікацію результуючої змінної.

Параметричними шарами мережі є перший і третій шари, а a_{ij}, b_{ij} і c_i є параметрами, які налаштовуються.

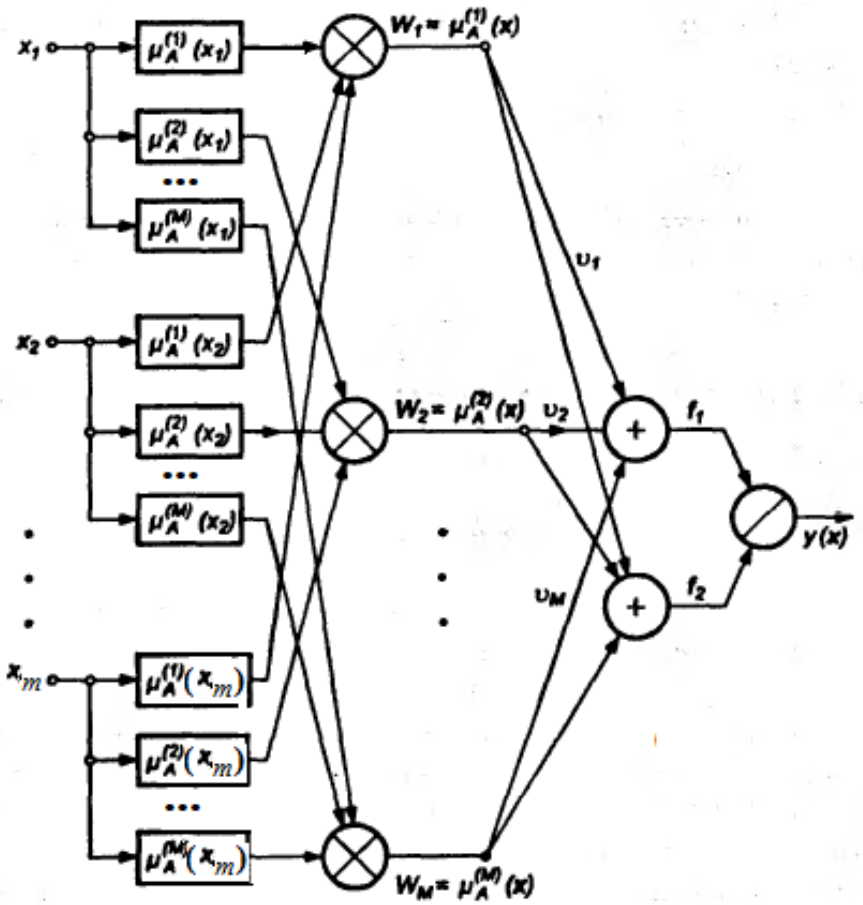


Рис.13.1 Структура нечіткої нейронної мережі Ванга-Менделя

Ця мережа має багатошарову структуру з прямим поширенням сигналу, значення виходу якої можна змінювати, корегуючи параметри елементів шарів, що дозволяє для навчання мережі використовувати алгоритм зворотного поширення похибки. В заданій для навчання вибірці ($x_1^{(k)}$, $x_2^{(k)}$, …, $x_m^{(k)}$, $y^{(k)}$), k=1, …, K, $x_1^{(k)}$, …, $x_m^{(k)}$ – значення вхідних змінних x_1, …, x_m, $y^{(k)}$ – еталонне значення вихідної змінної y в k-тому прикладі.

Алгоритм навчання складається з двох процедур.

Спочатку налаштовують лінійні параметри елементів третього шару c_i, а потім – параметри нелінійних ФН в елементах першого шару a_{ij} і b_{ij} (i=1, …,n; j=1, …,m). Розглянемо процедуру корегування значень c_i.

Процедура 1. *Крок 1*. Для кожного прикладу із навчальної вибірки за значеннями вхідних змінних $x_1^{(k)}$, …, $x_m^{(k)}$ (k = 1, …, K) обчислюють значення вихідних змінних $y'^{(k)}$.

Крок 2. Обчислюють функцію похибки для всіх прикладів навчальної вибірки: $E^{(k)} = \frac{1}{2}\left(y'^{(k)} - y^{(k)}\right)^2;\ k = 1,\ldots, \text{K}.$

Крок 3. Корегують значення c_i для кожного i-ого правила по кожному k-ому прикладу навчальної вибірки, за правилом: $c_{ij}(t+1) = c_{ij}(t) - \eta \frac{\partial E^{(k)}(t)}{\partial c_i(t)}$, $i=1,\ldots, n;\ k = 1,\ldots, \text{K}$, де t – номер ітерації навчання, $\eta \in [0,\ 1]$ – коефіцієнт швидкості навчання. Опускаючи проміжні перетворення, отримаємо:

$$c_i(t+1) = c_i(t) - \eta \frac{\left(y'^{(k)} - y^{(k)}\right) \prod_{j=1}^{m} \exp\left(-\left(\frac{x_j'^{(k)} - a_{ij}}{b_{ij}}\right)^2\right)}{\sum_{i=1}^{n} \prod_{j=1}^{m} \exp\left(-\left(\frac{x_j'^{(k)} - a_{ij}}{b_{ij}}\right)^2\right)}.$$

Процедура корегування значень c_i (етапи 1–3) ітераційно повторюється і вважається завершеною, якщо виконується одна із умов: 1) значення функції похибки на кожному прикладі навчальної вибірки не перевищує деякого встановленого порогу: $E^{(k)} < \varepsilon,\ k = 1,\ \ldots,\ \text{K};$ 2) оцінка середньої сумарної похибки нечіткої моделі з урахуванням всіх прикладів навчальної вибірки не перевищує деякого встановленого порогу: $E = \frac{1}{K}\sum_{k=1}^{K}\left(y'^{(k)} - y^{(k)}\right)^2 < \varepsilon;$ 3) похибка стабілізувалась на деякому значенні, яке « > ε». У перших двох випадках вважається, що нечітка мережа успішно навчилась, у третьому випадку переходять до процедури налаштування параметрів нелінійних ФН a_{ij} і b_{ij} в елементах першого шару.

Процедура 2. При виконанні процедури корегування значень a_{ij} і b_{ij} в елементах першого шару кроки 1, 2 співпадають із крокам процедури корегування значень c_i. На заключному етапі процедури значення a_{ij} і b_{ij} змінюються за такими правилами:

$$a_{ij}(t+1) = a_{ij}(t) - \eta \frac{\partial E^{(k)}(t)}{\partial a_i(t)} =$$

$$= a_{ij}(t) - \eta \frac{2\left(x_j'^{(k)} - a_{ij}\right)\left(y'^{(k)} - y^{(k)}\right)\left(c_i - y'^{(k)}\right)\prod_{j=1}^{m} \exp\left(-\left(\frac{x_j'^{(k)} - a_{ij}}{b_{ij}}\right)^2\right)}{b_{ij}^2 \sum_{i=1}^{n}\prod_{j=1}^{m} \exp\left(-\left(\frac{x_j'^{(k)} - a_{ij}}{b_{ij}}\right)^2\right)};$$

$$b_{ij}(t+1) = b_{ij}(t) - \eta \frac{\partial E^{(k)}(t)}{\partial b_i(t)} =$$

$$= b_{ij}(t) - \eta \frac{2\left(x_j'^{(k)} - a_{ij}\right)^2\left(y'^{(k)} - y^{(k)}\right)\left(c_i - y'^{(k)}\right)\prod_{j=1}^{m} \exp\left(-\left(\frac{x_j'^{(k)} - a_{ij}}{b_{ij}}\right)^2\right)}{b_{ij}^2 \sum_{i=1}^{n}\prod_{j=1}^{m} \exp\left(-\left(\frac{x_j'^{(k)} - a_{ij}}{b_{ij}}\right)^2\right)}$$

Умови завершення налаштування значень a_{ij} і b_{ij} подібні c_i. У випадку невиконання першої або другої умови процес ітераційно повторюється починаючи з корегування c_i до тих пір, поки нечітка мережа не буде коректно навчена.

Тема 2. Моделі із використанням РБФ–мережі

15. Алгоритми побудови Дерев класифікації: алгоритм С4.5

Мета роботи: ознайомитися з принципами побудови Дерев класифікації на основі алгоритму С4.5. *Об'єкт дослідження*: Дерева класифікації С4.5.

Питання для опрацювання

1. Дерева класифікації С4.5 [15].

Постановка підзадачі. На основі навчальної вибірки сформувати Дерево класифікації, яке генерує функцію c: $\Re^n \to L$, де L – множина міток класів. Наприклад, для бінарного класифікатора маємо $L=\{0, 1\}$. Якщо на вхід цього Дерева подається вектор $\mathbf{x}=\mathbf{x}^i=[x_1 \ ... \ x_n]^T$, $i=1,..., Q$, то його вихід $c(\mathbf{x})$ дорівнюватиме значенню «1», якщо мітка класу більше або дорівнює значенню «0,5», та «0», якщо мітка класу менше «0,5».

1. Завдання до практичної роботи

1. Вивчити теоретичний матеріал.

2. Сформувати Дерево класифікації С4.5 розв'язків в середовищі: MatLab, Python.

2. Контрольні запитання та завдання

1. Опишіть алгоритм формування Дерева класифікації С4.5.

Задачі для самостійного розв'язання: комп'ютерний практикум № 7

Аудиторна робота

Приклад 15.1. Класифікація на основі алгоритму ID3: визначення міри вибору атрибута *gain*.

Нехай задано табл. 15.1 – множину кортежів, які описують чи купив комп'ютер клієнт фірми. Визначити значення gain(дохід клієнта).

Розв'язання. Щоб визначити значення gain (**дохід_клієнта**)

Крок 1. Визначимо значення

Info (D)= $- 9/14 \cdot \log_2(9/14) - 5/14 \cdot \log_2(5/14) = 0.94$ bits;

Крок 2. Визначимо значення

$\text{Info}_{\text{дохід клієнта}}(D) =$

$4/14 \cdot (- 2/4 \cdot \log_2(2/4) - 2/4 \cdot \log_2(2/4)) +$ % високий

$+ 6/14 \cdot (- 4/6 \cdot \log_2(4/6) - 2/6 \cdot \log_2(2/6)) +$ % середній

$+4/14 \cdot (-3/4 \cdot \log_2(3/4) - 1/4 \cdot \log_2(1/4))$ % низький

=0.911 bits

Крок 3. Визначимо значення gain:

Gain(дохід клієнта)= Info (D) – $\text{Info}_{\text{дохід клієнта}}$ (D)=0.029 bits

Приклад 15.2. Класифікація на основі алгоритму С4.5: визначення міри вибору атрибута *GainRatio.*

Нехай задана табл. 15.1. Визначити значення GainRatio(дохід).

Таблиця 15.1

Множина кортежів **D** *з бази даних клієнтів компанії AllElectronics*

№ п.	Вік клієнта	Дохід клієнта	Чи є клієнт студентом	Кредитний рейтинг клієнта	Чи купив клієнт комп'ютер
1	молодий	великий	-	гарний	-
2	молодий	великий	-	дуже гарний	-
3	середній	великий	-	гарний	+
4	похилий	середній	-	гарний	+
5	похилий	невеликий	+	гарний	+
6	похилий	невеликий	+	дуже гарний	-
7	середній	невеликий	+	дуже гарний	+
8	молодий	середній	-	гарний	-
9	молодий	невеликий	+	гарний	+
10	похилий	середній	+	гарний	+
11	молодий	середній	+	дуже гарний	+
12	середній	середній	-	дуже гарний	+
13	середній	великий	+	гарний	+
14	похилий	середній	-	дуже гарний	-

Розв'язання. Приклади по атрибуту **дохід** розбивають дані таблиці 4 на три групи: низький, середній, високий, які містять 4, 6 і 4 кортежі, відповідно. Щоб обчислити коефіцієнт посилення атрибута **дохід**, спочатку розрахуємо значення:

$$SplitInfo_A(D) = -\frac{4}{14}\log_2\left(\frac{4}{14}\right) - \frac{6}{14}\log_2\left(\frac{6}{14}\right) - \frac{4}{14}\log_2\left(\frac{4}{14}\right) = 0.926.$$

Потім визначимо Gain(дохід).

Кроки 1, 2, 3. Приклад 15.1.

Крок 4. Визначимо значення GainRatio:

GainRatio (дохід) = 0.029/0.926 = 0.031.

Приклад 15.3. Розробка Дерев розв'язків в середовищі MatLab.

1. Формування класифікатора на основі алгоритму ID3: визначення міри вибору атрибута *gain*. Нехай задана табл. 15.1 – множина кортежів, що описують чи купив комп'ютер клієнт фірми. Побудувати Дерево розв'язків на основі вибору атрибута *gain*.

Таблиця 15.2

Множина кортежів D з бази даних клієнтів компанії AllElectronics

№ п.	Вік клієнта	Дохід клієнта	Чи є клієнт студентом	Кредитний рейтинг клієнта	Чи купив клієнт комп'ютер
1	2	3	4	5	6
1	Молодий-1	Великий-3	- 2	Гарний -1	-
2	Молодий -1	Великий-3	- 2	дуже гарний-2	-
3	Середній – 2	Великий-3	- 2	Гарний-1	+
4	Похилий -3	Середній-2	- 2	Гарний-1	+
5	Похилий -3	Невеликий-1	+ 1	Гарний-1	+
6	Похилий – 3	Невеликий-1	+ 1	дуже гарний-2	-

1	2	3	4	5	6
7	Середній – 2	Невеликий-1	+ 1	дуже гарний-2	+
8	Молодий – 1	Середній -2	- 2	Гарний-1	-
9	Молодий – 1	Невеликий-1	+ 1	Гарний-1	+
10	Похилий- 3	Середній-2	+ 1	Гарний-1	+
11	Молодий -1	Середній-2	+ 1	дуже гарний-2	+
12	Середній -2	Середній-2	- 2	дуже гарний-2	+
13	Середній – 2	Великий-3	+ 1	Гарний-1	+
14	Похилий - 3	Середній-2	- 2	дуже гарний-2	-

Перетволрюємо БД до вигляду:

№ п.	Вік клієнта	Дохід клієнта	Чи є клієнт студентом	Кредитний рейтинг клієнта	Чи купив клієнт комп'ютер
1	1	3	1	1	**0**
2	1	3	1	2	**0**
3	2	3	1	1	**1**
4	3	2	1	1	**1**
5	3	1	0	1	**1**
6	3	1	0	2	**0**
7	2	1	0	2	**1**
8	1	2	1	1	**0**
9	1	1	0	1	**1**
10	3	2	0	1	**1**
11	1	2	0	2	**1**
12	2	2	1	2	**1**
13	2	3	0	1	**1**
14	3	2	1	2	**0**

data.dat:

1 3 1 1 **0**
1 3 1 2 **0**
2 3 1 1 **1**
3 2 1 1 **1**
3 1 0 1 **1**
3 1 0 2 **0**
2 1 0 2 **1**
1 2 1 1 **0**
1 1 0 1 **1**
3 2 0 1 **1**
1 2 0 2 **1**
2 2 1 2 **1**
2 3 0 1 **1**
3 2 1 2 **0**

ID3 - main

```
clc; load data.dat; disp('Clients');
cols_attr(1, 1, 1) = {'Client age'};
cols_attr(1, 2, 1) = {' Young'};
cols_attr(1, 2, 2) = {1};
cols_attr(1, 3, 1) = {' Mean'};
cols_attr(1, 3, 2) = {2};
cols_attr(1, 4, 1) = {' Old'};
cols_attr(1, 4, 2) = {3};

cols_attr(2, 1, 1) = {'Income'};
```

```
    cols_attr(2, 2, 1) = {' Small'};
    cols_attr(2, 2, 2) = {1};
    cols_attr(2, 3, 1) = {' Normal'};
    cols_attr(2, 3, 2) = {2};
    cols_attr(2, 4, 1) = {' Big'};
    cols_attr(2, 4, 2) = {3};

    cols_attr(3, 1, 1) = {'Student?'};
    cols_attr(3, 2, 1) = {' Not student'};
    cols_attr(3, 2, 2) = {0};
    cols_attr(3, 3, 1) = {' Student'};
    cols_attr(3, 3, 2) = {1};
    cols_attr(4, 1, 1) = {'Credit'};
    cols_attr(4, 2, 1) = {' Good'};
    cols_attr(4, 2, 2) = {1};
    cols_attr(4, 3, 1) = {' Very good'};
    cols_attr(4, 3, 2) = {2};

    cols_attr(5, 1, 1) = {'Is buy computer?'};
    cols_attr(5, 2, 1) = {' No'};
    cols_attr(5, 2, 2) = {0};
    cols_attr(5, 3, 1) = {' Yes'};
    cols_attr(5, 3, 2) = {1};
    [tree, counter] = treeBuild(data, cols_attr, 0, 1);
```

Entropy

```
function Ip = entropy(data, column_number)
 % @data - all data; @column_number - number of column
 [rows, cols] = size(data);
 if rows == 0
     Ip = 0
 else
  unique_col_data = unique(data(:, column_number));
  Ip = 0;
  for j = 1:length(unique_col_data)
   pn=length(find(data(:,column_number)==unique_col_data(j)))     /
rows;
   Ip = Ip - pn * log2(pn);
  end;
end; end
```

Gain

```
function InfoT=Gain(data, column_number_attr, column_number)
      % @data - all data
      % @column_number_attr - column number
      % @column_number - number of column

    unique_col_data = unique(data(:, column_number_attr));
         [rows, cols] = size(data);
         InfoXT = 0;
         for i = 1:length(unique_col_data)
```

```
                find_data  =  find(data(:,  column_number_attr)  ==
unique_col_data(i));
                attr_data = data(find_data, :);
                InfoXT                                              =
InfoXT+(length(find_data)/rows)*entropy(attr_data, column_number);
            end;
            InfoT = entropy(data,column_number) - InfoXT;
        End
```

treeBuild

```
function [tree,counter]=treeBuild(data, cols_attr, counter, level,
tree)
      % @data - matrix with data
      % @cols_attr - string array with columns and colums unique
parameters names
      % @counter - integer value with a last value in tree
      % @level - save level number of tree
      % @tree - save tree recurtion arguments
      if nargin < 5
        tree = [];
      end;
      counter = counter + 1;
      [rows, cols] = size(data);
      tree_gains = [];
      for i = 1:cols - 1 % get Gain params
          tree_gain(i, 1) = i; % column number
          tree_gain(i, 2)=Gain(data,i,cols); %Gain coefficient
      end;
      % max_gain = tree_gain(find(max(tree_gain(:, 2))), :);
      % Find maximum Gain value in get results
      max_gain=tree_gain(find(max(tree_gain(:,2))==
tree_gain(:,2)), :);
     % Find maximum Gain value in get results
      for k = 2:length(cols_attr(max_gain(1), :, :))
       if (isempty(cell2mat(cols_attr(max_gain(1),k,1))) == 0)
     disp(cell2mat(strcat(buildTreeLevel(level),cols_attr(max_gain(
1),k,1))));
          this_data=data(find(data(:,        max_gain(1))        ==
cell2mat(cols_attr(max_gain(1), k, 2))), :); % get data for this
node
          [rows,cols]=size(this_data); % get new data matrix size
          this_data=this_data(:, max_gain(1) + 1:cols);
% get all columns in @this_data after @max_gain column
          this_cols_attr = cols_attr(max_gain(1) + 1:cols, :, :); %
get all columns in @cols_attr after @max_gain column
          [rows, cols]=size(this_data); % get new data matrix size
after shortcut parameter @this_data

       if entropy(this_data(:, cols), 1) == 0
        temp = this_data(:, cols);
        counter = counter + 1;
```

```
        if temp(1) == 1
        disp(strcat(buildTreeLevel(level + 1), ' Success'))
        elseif temp(1) == 0
        disp(strcat(buildTreeLevel(level + 1), ' Failer'))
        end;
        counter = counter + 1;
        else
     [tree,counter]=treeBuild(this_data,this_cols_attr,    counter,
level + 1, tree);
       end;
      end;end;
```

buildTreeLevel

```
function level_lines = buildTreeLevel(count)
          level_lines = '';
          for i = 1:count
               level_lines = strcat('--', level_lines);
          end;
```

Результат

```
Clients
-- Young
---- Not student
------ Failer
---- Student
------ Success
-- Mean
---- Success
-- Old
---- Good
------ Success
---- Very good
------ Failer
```

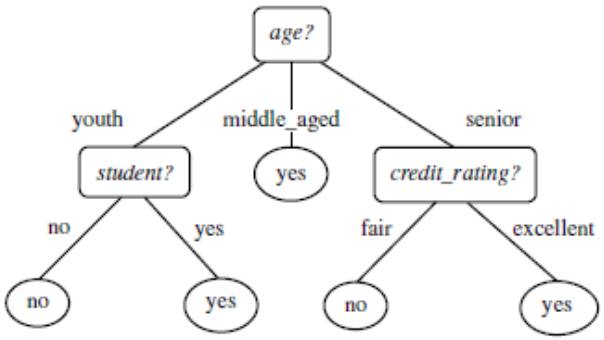


Рис. 15.1. Схема прийняття рішень для концепції покупки комп'ютера, яка показує, чи може клієнт в AllElectronics придбати комп'ютер

2. Формування класифікатора на основі алгоритму C4.5

C4.5 - main

```
clc; load data_1.dat;
disp('Clients');
cols_attr(1, 1, 1) = {'Client age'};
cols_attr(1, 2, 1) = {' Young'};
cols_attr(1, 2, 2) = {1};
cols_attr(1, 3, 1) = {' Mean'};
cols_attr(1, 3, 2) = {2};
cols_attr(1, 4, 1) = {' Old'};
cols_attr(1, 4, 2) = {3};

cols_attr(2, 1, 1) = {'Income'};
cols_attr(2, 2, 1) = {' Small'};
```

```
cols_attr(2, 2, 2) = {1};
cols_attr(2, 3, 1) = {' Normal'};
cols_attr(2, 3, 2) = {2};
cols_attr(2, 4, 1) = {' Big'};
cols_attr(2, 4, 2) = {3};

cols_attr(3, 1, 1) = {'Student?'};
cols_attr(3, 2, 1) = {' No'};
cols_attr(3, 2, 2) = {0};
cols_attr(3, 3, 1) = {' Yes'};
cols_attr(3, 3, 2) = {1};

cols_attr(4, 1, 1) = {'Credit'};
cols_attr(4, 2, 1) = {' Good'};
cols_attr(4, 2, 2) = {1};
cols_attr(4, 3, 1) = {' Very good'};
cols_attr(4, 3, 2) = {2};

cols_attr(5, 1, 1) = {'Is buy computer?'};
cols_attr(5, 2, 1) = {' No'};
cols_attr(5, 2, 2) = {0};
cols_attr(5, 3, 1) = {' Yes'};
cols_attr(5, 3, 2) = {1};
[tree, counter] = treeBuild(data_2, cols_attr, 0, 1);
```

Gain - визначаємо GainRatio

```
function GainRatio=Gain(data,column_number_attr, column_number)
 % отримуємо вхідні дані
 col_data = unique(data(:, column_number_attr));
 [rows, cols] = size(data);InfoX = 0;
 % розраховуємо InfoD) за класом
 for i = 1:length(col_data)
  find_data=find(data(:, column_number_attr) == col_data(i));
  attr_data = data(find_data, :);
  InfoX=InfoX+(length(find_data)/rows)*entropy(attr_data,
column_number);
 end;
 % знаходимо Gain
 Gain = entropy(data, column_number) - InfoX;
 % знаходимо GainRatio
 GainRatio = Gain / entropy(data, column_number);
end
```

рядки 2 – 4 представляют собой прием и обработку входных данных, запись их в переменные.

рядки 6 – 12 – визначення Info(D) (ентропії для всіх можливих типів атрибутів в класах)

рядок 14 – визначення Gain = Indo(D) - Info(D)

рядок 16 – розрахунок GainRatio для заданих класів

treeBuild.m – об’єднання: збірка всього Дерева

```
function [tree, counter]=treeBuild(data, cols_attr, counter,
level, tree)
     % @data - matrix with data
     % @cols_attr - string array with columns and colums unique
parameters names
     % @counter - integer value with a last value in tree
     % @level - save level number of tree
     % @tree - save tree recurtion arguments
if nargin < 5
 tree = [];
end;
counter=counter + 1; [rows, cols] = size(data);tree_gains=[];
for i = 1:cols - 1 % get Gain params
 tree_gain(i, 1) = i; % column number
 tree_gain(i, 2) = Gain(data, i, cols); % Gain coefficient
end;
max_gain=tree_gain(find(max(tree_gain(:, 2)) == tree_gain(:, 2)),
:);
% Find maximum Gain value in get results (c4.5 part of algorithm)
for k = 2:length(cols_attr(max_gain(1), :, :))
 if (isempty(cell2mat(cols_attr(max_gain(1), k, 1))) == 0)
     disp(cell2mat(strcat(buildTreeLevel(level),cols_attr(max_gain(
1), k, 1))));
     this_data=data(find(data(:, max_gain(1)) ==
cell2mat(cols_attr(max_gain(1), k, 2))), :); % get data for this
node
     [rows, cols] = size(this_data); % get new data matrix size
     this_data = this_data(:, max_gain(1) + 1:cols); % get all
columns in @this_data after @max_gain column
     this_cols_attr = cols_attr(max_gain(1) + 1:cols, :, :); % get
all columns in @cols_attr after @max_gain column
     [rows, cols] = size(this_data); % get new data matrix size
after shortcut parameter @this_data
     if entropy(this_data(:, cols), 1) == 0
temp = this_data(:, cols);
     counter = counter + 1;
     if temp(1) == 1
      disp(strcat(buildTreeLevel(level + 1), ' +'))
     elseif temp(1) == 0
      disp(strcat(buildTreeLevel(level + 1), ' -'))
     end;
     counter = counter + 1;
   else
     [tree,counter]=treeBuild(this_data, this_cols_attr, counter,
level + 1, tree);
end;end;end;
```

entropy.m – розрахунок ентропії за заданими параметрами

```
function Ip = entropy(data, column_number)
     % @data - all data;  @column_number - number of column
     [rows, cols] = size(data);
```

```
    if rows == 0
        Ip = 0
    else
        unique_col_data = unique(data(:, column_number));
    Ip = 0;
    for j = 1:length(unique_col_data)

    pn=length(find(data(:,column_number)==unique_col_data(j)))/
rows;
        Ip = Ip - pn * log2(pn);
end; end; end
```

buildTreeLevel.m – побудова одного рівня Дерева

```
function level_lines = buildTreeLevel(count)
    level_lines = '';
    for i = 1:count
        level_lines = strcat('--', level_lines);
    end;
```

Результат

```
    Clients
    -- Young
    ---- No
    ------ -
    ---- Yes
    ------ +
    -- Mean
    ---- +
    -- Old
    ---- Good
    ------ +
    ---- Very good
    ------ -
```

Теоретичний матеріал

Дерева класифікації C4.5 [15, 16] (алгоритм C4.5 є наступником ID3)

1. Дерева класифікації. Серед різних моделей класифікаторів відомі класифікатори на основі Дерева розв'язків. Дерево розв'язків є популярним методом аналізу даних, основою якого є навчання на прикладах і формування відповідної ієрархічної структури. Дерево розв'язків розбиває вхідний простір (відомий як простір атрибутів) множини даних на взаємовиключні області, кожній з яких присвоєно ім'я. Механізм прийняття рішень є прозорим, оскільки за деревоподібною структурою можна легко пояснити як приймається рішення.

Індукційне Дерево розв'язків є деревоподібною структурою, навченою на основі мічених класів з використанням множини прикладів навчання у вигляді

кортежів. В цій структурі кожний внутрішній (некінцевий) вузол зображує тестове значення атрибута з прикладу навчання, кожна гілка зображує приклад навчання моделі, кожен лист (або вузол–термінал) зображує мітку класу (вихід моделі). Кореневий вузол є найвищою вершиною в Дереві.

Дерево розв'язків може обробляти дані високої розмірності. У багатовимірних областях простору атрибутів одна гілка Дерева розв'язків зазвичай залучає тільки частину атрибутів.

Дерево розв'язків, яке використовують для розв'язання задач класифікації, називають *Деревом класифікації*, і кожен вузол-термінал містить мітку, яка вказує передбачуваний клас заданого вектора (прикладу) ознак. Щоб побудувати Дерево класифікації, необхідно мати похибку вимірювання $E(t)$, яка кількісно описує продуктивність вузла t при розбитті даних (прикладів) з різних класів. Похибку Дерев класифікації часто називають *функцією домішок заданого вузла*. Вона досягає мінімального значення, наприклад нуля, якщо всі дані належать до одного класу, і максимального, якщо дані розподілені рівномірно по всіх можливих класах.

У загальному випадку, Дерева класифікації мають прийнятну точність. Проте їхнє успішне застосування залежить від наявних даних.

Дерева класифікації формують на основі методу вибору атрибутів, який визначає атрибут, що «найкраще» розпізнає заданий кортеж відповідно до класу. Цей метод реалізовано у вигляді процедури, яка використовує **міру вибору атрибуту**. Критерій розбиття може вказувати на відповідний атрибут або на точки розбиття, або на відповідні підмножини даних.

Критерій розбиття. Більшість існуючих алгоритмів формування Дерев класифікації на основі даних використовують *рекурсивну процедуру*, яка на кожному кроці визначає критерій розбиття множини навчання на дві підмножини таким чином, щоб максимізувати їхню однорідність.

Критерій розбиття визначається таким чином, що в ідеалі в результаті розбиття кожний підрозділ даних є таким, що всі кортежи в ньому відносяться до одного класу (тобто кожен підрозділ є чистим).

Існують *три можливих сценарії розбиття* кортежів БД **D** за атрибутом. Нехай кортеж **x** подано у вигляді n–вимірного вектора атрибутів: $\mathbf{x} = [x_1 \ldots x_n]^T$, зображуючи n вимірювань з n атрибутами (A_1, ..., A_n). Кожен атрибут зображує «ознаку» **x**. Передбачається, що кожен кортеж **x** належить визначеному класу (атрибуту «мітка класу»). Атрибут мітки класу має дискретні невпорядковані значення (кожне значення служить класом). Кортежі, що формують множину навчання, називають кортежі навчання і вибираються з БД при аналізі.

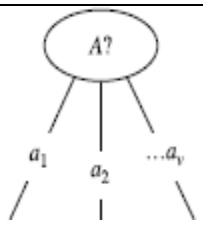
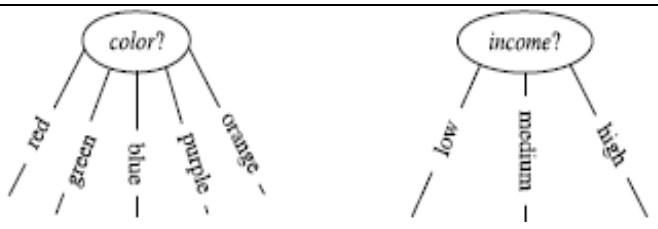
Нехай A є атрибутом розбиття, який може мати v різних значень $\{a_1, \ldots, a_v\}$, заснованих на даних навчання, тоді маємо: якщо значення

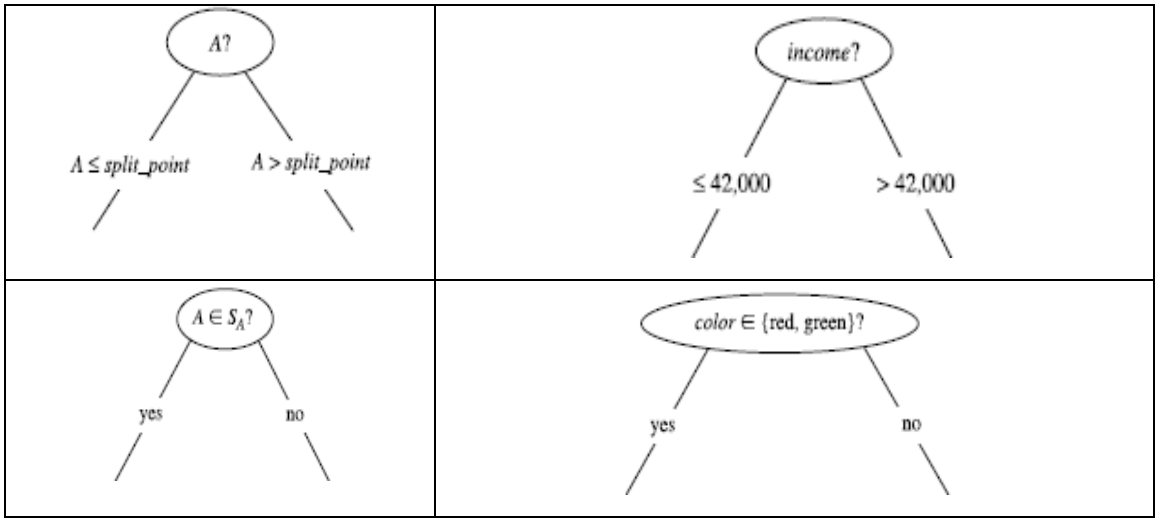
а) дискретно–значні, то гілка росте на кожному відомому значенні A,

б) неперервно–значні, то дві гілки ростуть відповідно до точки розбиття;

с) дискретно–значні і має бути сформовано бінарне дерево, то тест має вигляд «$A \in S_A$», де S_A – множина розбиття для A

Алгоритми побудови Дерев класифікації. Алгоритми ID3 і CART були розроблені незалежно один від одного приблизно в один і той же час, вони будують Дерева розв'язків шляхом вивчення кортежів даних. Алгоритми ID3, C4.5 і CART реалізують підхід, при якому Дерево будують рекурсивно зверху вниз, використовуючи такі **міри вибору атрибутів**, як приріст інформації Gain, коефіцієнт приросту інформації Gain Ratio, коефіцієнт Gini та інші.

Сценарії розбиття	*Приклади*
	



Алгоритми ID3, C4.5 і CART реалізують підхід, при якому Дерева будуються рекурсивно зверху вниз, використовуючи при цьому такі *міри вибору атрибутів*, як приріст інформації Gain, коефіцієнт приросту інформації Gain Ratio, коефіцієнт Gini та інші. На практиці ці міри дають досить хороші результати для багатозначних атрибутів:

1. Алгоритм C4.5 з коефіцієнтом приросту інформації Gain Ratio використовують при незбалансованому розбитті (коли один клас набагато менший за інші).

2. Алгоритм CART з коефіцієнтом Gini здатен визначити багатовимірне розбиття на основі лінійної комбінації атрибутів. Це розбиття є структурною формою атрибута, коли нові атрибути утворюють на основі вже існуючих.

2. Алгоритм ID3. Алгоритм ID3 в якості міри вибору атрибутів використовує *приріст інформації gain*. Ця міра заснована на роботі Клода Шеннона з теорії інформації, який вивчав «інформаційний зміст» повідомлення.

Нехай вузол N містить кортежі розділу даних **D**. Для виконання розбиття у вузлі дерева N вибирають атрибут з найбільшим значення приросту інформації. Цей атрибут відображає міру «домішків» у сформованих підрозділах **D** після розбиття.

Очікувана інформація, необхідна для класифікації кортежів з **D**, задається у вигляді: $Info(\mathbf{D}) = -\sum_{i=1}^{m} p_i \log_2(p_i)$, де *Info(***D***)* – ентропія **D**, p_i – ймовірність того, що довільний набір кортежів з **D** відноситься до класу C_i (i=1, …, m) і оцінюється значенням $|C_{i,\mathbf{D}}|/|\mathbf{D}|$. Інформація кодується в бітах, тому використовується функція log2. Значення Info(**D**) –це середня кількість інформації, необхідна для визначення мітки класу кортежів з **D**.

Необхідно розбити кортежі з розділу даних **D** на основі деякого атрибута A, який може мати V різних значень $\{a_1, ..., a_v\}$. Атрибут A може бути використаний для розбиття **D** на V підрозділи $\{\mathbf{D}_1, ..., \mathbf{D}_V\}$, де $\mathbf{D}_j$ містить ті кортежі з **D**, які мають результат a_j з A. Ці підрозділи будуть відповідати гілці, яка виросла з вузла *N*.

В ідеалі необхідно мати розбиття, яке реалізує точну класифікацію кортежів (щоб кожен підрозділ був чистим). Однак, ймовірно, що підрозділ буде нечистим (наприклад, коли підрозділ містить набір з кортежів, що належать різним класам). Скільки додаткової інформації нам потрібно після розбиття, щоб прийти до точної класифікації? Ця кількість вимірюється за формулою: $Info_A(\mathbf{D}) = \sum_{j=1}^{v} \frac{|\mathbf{D}_j|}{|\mathbf{D}|} Info(\mathbf{D}_j)$, де елемент $|\mathbf{D}_j|/|\mathbf{D}|$ виступає в якості ваги *j*-ого розділу, $Info_A(\mathbf{D})$ – очікувана інформація, необхідна для класифікації кортежів з **D** на основі розбиття A. Чим менше значення очікуваної інформації потрібно, тим більше чистота розділів.

Приріст інформації визначається як різниця між первісною необхідною інформацією (тобто інформацією, заснованою тільки на пропорції класів) і новими вимогами (наприклад, інформацією, отриманою після розбиття на основі атрибуту A): $Gain(A)=Info(\mathbf{D}) - Info_A(\mathbf{D})$.

Gain(A) говорить нам, скільки інформації буде отримано шляхом розгалуження Дерева на основі атрибута A. Атрибут A з найбільшим посиленням інформації Gain(A) було обрано в якості атрибута розбиття у вузлі

N. Розбиття на основі атрибута A буде виконувати «найкращу класифікацію», якщо значення $\text{Info}_A(\mathbf{D})$ буде мінімальним.

3. Алгоритм C4.5. Міра приросту інформації Gain віддає перевагу атрибутам, які мають велику кількість значень. Алгоритм C4.5 використовує розширення приросту інформації, відоме як *коефіцієнт посилення GainRatio*. Цей алгоритм застосовує певного роду нормалізацію приросту інформації використовуючи значення розбиття інформації $\text{SplitInfo}_A(\mathbf{D})$: $SplitInfo_A(\mathbf{D}) = -\sum_{j=1}^{v} \frac{|\mathbf{D}_j|}{|\mathbf{D}|} \log_2 \left(\frac{|\mathbf{D}_j|}{|\mathbf{D}|} \right)$. Він зображує інформацію, отриману шляхом розбиття набору даних навчання з **D** на V розділи, які відповідають V результатам перевірки атрибута A.

Коефіцієнт посилення визначається за формулою: $GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(\mathbf{D})}$. Атрибут з максимальним коефіцієнтом посилення є обраним в якості атрибута розбиття.

Коли інформація, отримана при розбитті на основі атрибута A, є максимальною (тобто $Info_A(\mathbf{D})$ прагне до значення «0»), співвідношення стає нестійким. Щоб уникнути цього, додається обмеження, в результаті чого тест на приріст інформації повинен бути великим (наприклад, настільки великий, як середній приріст за всіма розглянутими тестам).

16. Алгоритми побудови Дерев класифікації: алгоритм CART.

Мета роботи: ознайомитися з принципами побудови Дерев класифікації на основі алгоритму CART. *Об'єкт дослідження*: Дерева класифікації CART.

Питання для опрацювання

1. Дерева класифікації CART [15].

Постановка підзадачі. На основі навчальної вибірки сформувати Дерево класифікації, яке генерує функцію c: $\Re^n \to L$, де L – множина міток класів. Наприклад, для бінарного класифікатора маємо $L=\{0, 1\}$. Якщо на вхід цього Дерева подається вектор $\mathbf{x}=\mathbf{x}^i=[x_1 \ ... \ x_n]^T$, $i=1,..., Q$, то його вихід $c(\mathbf{x})$ дорівнюватиме значенню «1», якщо мітка класу більше або дорівнює значенню «0,5», та «0», якщо мітка класу менше «0,5».

1. Завдання до практичної роботи

1. Вивчити теоретичний матеріал.
2. Підготувати дані навчання та тестування для дослідження.

Реалізувати відповідний додаток Побудови Дерева класифікації.

3. Сформувати Дерево класифікації CART в середовищі: MatLab, Python.

2. Контрольні запитання та завдання

1. Опишіть алгоритм формування Дерева класифікації CART.

Задачі для самостійного розв'язання: комп'ютерний практикум № 7

Аудиторна робота

Приклад 16.1. Класифікація на основі алгоритму CART: визначення міри вибору атрибута з використанням индексу Джіні (Gini).

Нехай задана табл. 15.1 (розділ даних **D**). На основі алгоритму CART (значення індексу Джіні) визначити кореневий вузол.

*Розв'язання.*Маємо **D** – підготовлені дані з табл 15.1, де є дев'ять кортежів, які належать класу «*купляє комп'ютер*="так"», а решта п'ять кортежів належать до класу «*купляє комп'ютер*="ні"». Для кортежів із **D** сформуємо вузол N (корінь).

Крок 1. Визначимо значення індексу Джіні домішок у **D**:

$$Gini(\mathbf{D}) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0,459.$$

Крок 2. Щоб визначити критерій розбиття кортежій із **D**, необхідно обчислити індекс Джіні для кожного атрибуту.

Розпочнемо з атрибуту **дохід** і розглянемо кожну із можливих підмножин розбиття. Розглянемо підмножини {low-невеликий, medium-середній} і {*high*-великий}. Маємо 10 кортежів в розділі $\mathbf{D}_1$, які задовольняють умову дохід$\in$\{*low*, *medium*\}; решта 4 кортежи із **D** передані до розділу $\mathbf{D}_2$: дохід$\in$(*high*). Значення індексу Джіні на основі цього розбиття обчислимо таким чином:

$$Gini_{\text{дохід}\in\{low,medium\}}(\mathbf{D}) = \frac{10}{14}Gini(\mathbf{D}_1) + \frac{4}{14}Gini(\mathbf{D}_2) =$$

$$= \frac{10}{14}\left(1-\left(\frac{7}{10}\right)^2-\left(\frac{3}{10}\right)^2\right)+\frac{4}{14}\left(1-\left(\frac{2}{4}\right)^2-\left(\frac{2}{4}\right)^2\right) = 0.443 = Gini_{\text{дохід}\in\{high\}}(\mathbf{D}).$$

При іншому розбитті на основі значень атрибуту **дохід** (на підмножини {low, high} і {medium}, {medium, high} і {low}) значення індексу Джіні дорівнює:

$$Gini_{\text{дохід}\in\{low,high\}}(\mathbf{D}) = \frac{8}{14}Gini(\mathbf{D}_1) + \frac{6}{14}Gini(\mathbf{D}_2) =$$

$$= \frac{8}{14}\left(1-\left(\frac{5}{8}\right)^2-\left(\frac{3}{8}\right)^2\right)+\frac{6}{14}\left(1-\left(\frac{4}{6}\right)^2-\left(\frac{2}{6}\right)^2\right) = 0.857 = Gini_{\text{дохід}\in\{medium\}}(\mathbf{D}).$$

$Gini_{\text{дохід}\in\{\text{low, high}\}}(\mathbf{D})$ = 0,857 для підмножин {low, high} і {medium};

$$Gini_{\text{дохід}\in\{medium,high\}}(\mathbf{D}) = \frac{10}{14}Gini(\mathbf{D}_1) + \frac{4}{14}Gini(\mathbf{D}_2) =$$

$$= \frac{10}{14}\left(1-\left(\frac{6}{10}\right)^2-\left(\frac{4}{10}\right)^2\right)+\frac{4}{14}\left(1-\left(\frac{3}{4}\right)^2-\left(\frac{1}{4}\right)^2\right) = 0.557 = Gini_{\text{дохід}\in\{low\}}(\mathbf{D}).$$

$Gini_{\text{дохід}\in\{\text{medium, high}\}}(\mathbf{D})$ = 0,557 для підмножин {medium, high} і {low}.

Мінімальним є значення «0,443», тому найкращим бінарним розбиттям для атрибуту **дохід** є {low, medium} і {high} (це приводить до мінімуму індекс Джіні $Gini_{\text{A}}(\mathbf{D})$).

Оцінюючи атрибут **вік**, отримаємо {youth, senior} і {middle_aged} як найкраще розбиття за цим атрибутом з індексом Джіні, який дорівнює значенню 0,357.

$$Gini_{\text{вік}\in\{мол,сер\}}(\mathbf{D}) = \frac{9}{14}Gini(\mathbf{D}_1) + \frac{5}{14}Gini(\mathbf{D}_2) =$$

$$= \frac{9}{14}\left(1-\left(\frac{6}{9}\right)^2-\left(\frac{3}{9}\right)^2\right)+\frac{5}{14}\left(1-\left(\frac{3}{5}\right)^2-\left(\frac{2}{5}\right)^2\right) = 0.857 = Gini_{\text{вік}\in\{пох\}}(\mathbf{D}).$$

$$Gini_{\text{вік}\in\{пох,сер\}}(\mathbf{D}) = \frac{9}{14}Gini(\mathbf{D}_1) + \frac{5}{14}Gini(\mathbf{D}_2) =$$

$$= \frac{9}{14}\left(1-\left(\frac{7}{9}\right)^2-\left(\frac{2}{9}\right)^2\right)+\frac{5}{14}\left(1-\left(\frac{2}{5}\right)^2-\left(\frac{3}{5}\right)^2\right) = 0.857 = Gini_{\text{вік}\in\{мол\}}(\mathbf{D}).$$

$$Gini_{\text{вік}\in\{мол,пох\}}(\mathbf{D}) = \frac{10}{14}Gini(\mathbf{D}_1) + \frac{4}{14}Gini(\mathbf{D}_2) =$$

$$= \frac{10}{14}\left(1-\left(\frac{5}{10}\right)^2-\left(\frac{5}{10}\right)^2\right)+\frac{4}{14}\left(1-\left(\frac{4}{4}\right)^2-\left(\frac{0}{4}\right)^2\right) = 0.3571 = Gini_{\text{вік}\in\{сер\}}(\mathbf{D}).$$

Атрибути **студент** і **кредитний рейтинг** є бінарними зі значеннями індексу Джіні $Gini_{\text{A}}(\mathbf{D})$ 0,857 і 0,429, відповідно:

$$Gini_{\text{студент}\in\{+\}}(\mathbf{D}) = \frac{7}{14}Gini(\mathbf{D}_1) + \frac{7}{14}Gini(\mathbf{D}_2) =$$

$$= \frac{7}{14}\left(1-\left(\frac{6}{7}\right)^2-\left(\frac{1}{7}\right)^2\right)+\frac{7}{14}\left(1-\left(\frac{3}{7}\right)^2-\left(\frac{4}{7}\right)^2\right) = 0.8571 = Gini_{\text{студент}\in\{-\}}(\mathbf{D}).$$

$$Gini_{\text{кр.рейтинг}\in\{\textit{гарний}\}}(\mathbf{D}) = \frac{8}{14}Gini(\mathbf{D}_1) + \frac{6}{14}Gini(\mathbf{D}_2) =$$

$$= \frac{8}{14}\left(1-\left(\frac{6}{8}\right)^2-\left(\frac{2}{8}\right)^2\right)+\frac{6}{14}\left(1-\left(\frac{3}{6}\right)^2-\left(\frac{3}{6}\right)^2\right) = 0.714 = Gini_{\text{кр.рейтинг}\in\{\textit{дуже гарн}\}}(\mathbf{D}).$$

Тому атрибут **вік** і підмножина розбиття {*молод.*, *похол.*} і {середн.} в цілому дають мінімальне значення індексу Джіні $Gini_A(\mathbf{D})$ із зниженням домішок: ΔGini = 0,459 – 0,357. Це бінарне розбиття призводить до максимального зниження забрудненості кортежів у **D**. Вузол *N* позначений критерієм **вік**, дві гілки ростуть з нього, кортежи розбиті відповідно. Таким чином, індекс Джіні обрав атрибут **вік.**

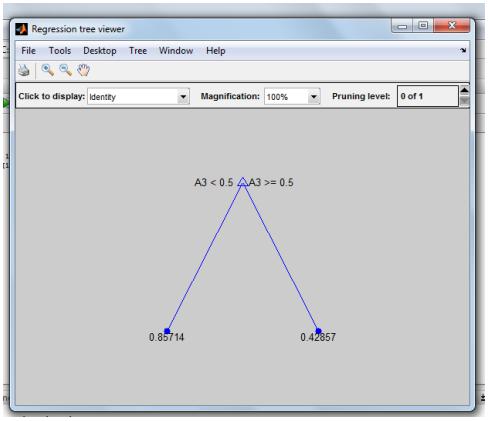
Зниження домішок, які можуть виникнути на основі бінарного розбиття на дискретному значенні атрибуту A визначають за формулою: $\Delta Gini(A) = Gini(D) - Gini_A(D)$. Атрибут, який максимізує скорочення домішки (або має мінімальний індекс Джині), обирають за точку розбиття атрибута.

data.dat:

1 3 1 1 **0**	1 2 1 1 **0**
1 3 1 2 **0**	1 1 0 1 **1**
2 3 1 1 **1**	3 2 0 1 **1**
3 2 1 1 **1**	1 2 0 2 **1**
3 1 0 1 **1**	2 2 1 2 **1**
3 1 0 2 **0**	2 3 0 1 **1**
2 1 0 2 **1**	3 2 1 2 **0**

Приклад 16.2: MatLab. Побудова Дерева CART в середовищі MatLab.

```
clear all;close all;clc;
data=[
[1, 3, 1, 1, 0];[1, 3, 1, 2, 0];[2, 3, 1, 1, 1];
[3, 2, 1, 1, 1];[3, 1, 0, 1, 1];[3, 1, 0, 2, 0];
[2, 1, 0, 2, 1]; [1, 2, 1, 1,0]; [1, 1, 0, 1, 1];
[3, 2, 0, 1, 1]; [1, 2, 0, 2, 1]; [2, 2, 1, 2, 1];
[2, 3, 0, 1, 1];[3, 2, 1, 2, 0]]
x=data(:,[1:4]); cl=data(:,5);
t=classregtree(x,cl,'names',{'A1' 'A2' 'A3' 'A4'}); view(t)
```




```
clear all;close all;clc;
fID=fopen('data1.txt','r');
data=fscanf(fID,'%f,%f,%f,%f,%f,%f,%f,%f, %f', [9 inf])';
data=data([1:381],:);
x=data(:,[1:8]); cl=data(:,9);
t=classregtree(x,cl,'names',{'A1' 'A2' 'A3' 'A4' 'A5' 'A6' 'A7'
'A8'}); view(t)

data =
0.1100 0.1000 0.1000 0
0.1200 0.1000 0.1000 0
0.2250 0.6500 0.3000 1.0000
0.2350 0.6500 0.3000 1.0000
0.4500 0.2500 0.5000 0
0.6700 0.8500 0.7000 1.0000
0.7000 0.8500 0.7000 1.0000
0.8800 0.4600 0.9000 1.0000
0.9900 0.4600 0.9000 1.0000

clear all;close all;clc;
data=[
 [0.1100,0.1000, 0.1,   0];[0.1200,0.1000, 0.1,   0]
 [0.2250,0.6500, 0.3,   1];[0.2350,0.6500, 0.3,   1]
 [0.4500,0.2500, 0.5,   0];[0.6700,0.8500, 0.7,   1]
 [0.700, 0.8500, 0.7,   1];[0.8800,0.4600, 0.9,   1]
 [0.99,  0.4600, 0.9,   1]]
x=data(:,[1:3]); cl=data(:,4);
t = classregtree(x,cl,'names',{'A1' 'A2' 'A3'});view(t)
t = treeval(x,cl,'names',{'A1' 'A2' 'A3'});
t = treedisp(x,cl,'names',{'A1' 'A2' 'A3'});
view(t,varargin{:});
```

Для нової версії MatLab маємо

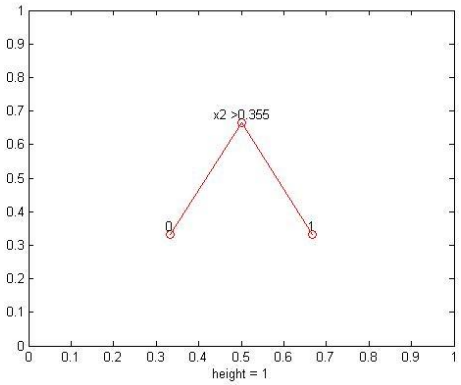
```
clear all;close all;clc;
load ionosphere
tc=fitctree(X, Y); view(tc)
mdlDefault=fitctree(X,Y,'CrossVal','on');
view (mdlDefault.Trained{1}, 'Mode', 'graph')
```

Приклад 16.3. Побудова Дерева CART в середовищі MatLab

```
Data.txt
 0.1100,0.1000, 0.1,   1
 0.1200,0.1000, 0.1,   1
 0.2250,0.6500, 0.3,   2
 0.2350,0.6500, 0.3,   2
 0.4500,0.2500, 0.5,   1
 0.6700,0.8500, 0.7,   2
 0.700, 0.8500, 0.7,   2
 0.8800,0.4600, 0.9,   2
 0.99,  0.4600, 0.9,   2
```

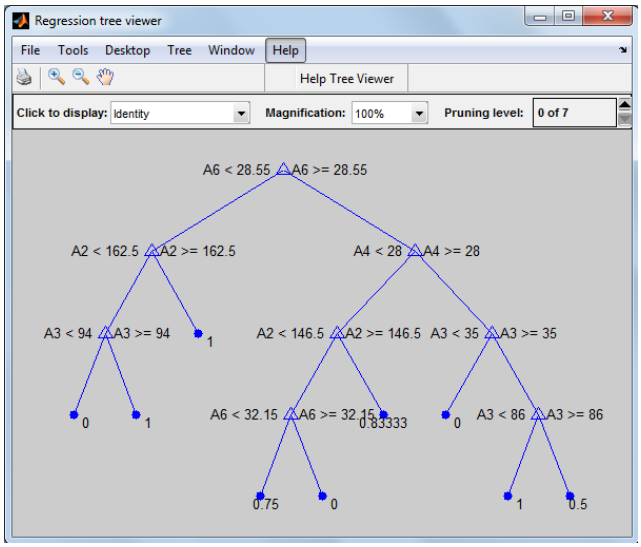
X2: 0.46-0.25=0.21 /2 =0.105 → 0.25+0.105=**0.355**
(0.46+0.25)/2=**0.355**

```
clear all;close all;clc;
fID = fopen('data.txt','r');
data.X = fscanf(fID,'%f, %f, %f, %f', [4 inf])';
x=data.X(:,[1:2]); cl=data.X(:,4);
%t = classregtree(x,cl);view(t)
ctree = ClassificationTree.fit(x,cl);view(ctree)
resuberror = resubLoss(ctree)
 1  class = 2
 resuberror =0.3333
 ctree = ClassificationTree:
    PredictorNames:{'x1'  'x2'}
 CategoricalPredictors: []
  ResponseName: 'Y'
    ClassNames: [1 2]
    ScoreTransform: 'none'
 NObservations: 9
```



DATA2 – 50 элементов (Додаток В)

```
clear all;close all;clc; fID = fopen('data2.txt','r');
data=fscanf(fID,'%f,%f,%f,%f,%f,%f,%f,%f,%f',[9 inf])';
data=data([1:50],:); x=data(:,[1:8]); cl=data(:,9);
t = classregtree(x,cl,'names',{'A1' 'A2' 'A3' 'A4' 'A5' 'A6' 'A7'
'A8'}); view(t)
```



Decision tree for classification

```
1  if x6<28.55 then node 2 elseif x6>=28.55 then node 3 else 1
2  if x2<162.5 then node 4 elseif x2>=162.5 then node 5 else 0
3  if x4<28 then node 6 elseif x4>=28 then node 7 else 1
4  if x3<94 then node 8 elseif x3>=94 then node 9 else 0
5  class = 1
6  if x2<146.5 then node 10 elseif x2>=146.5 then node 11 else 0
7  if x3<35 then node 12 elseif x3>=35 then node 13 else 1
8  class = 0
9  class = 1
10  if x6<32.15 then node 14 elseif x6>=32.15 then node 15 else 0
11  class = 1
12  class = 0
13  class = 1
14  class = 1
15  class = 0
resuberror =  0.0600
```

Теоретичний матеріал

Дерева класифікації CART [15, 16]

1. Метод CART: *індекс Джині (gini).* Індекс Джині розглядає бінарне розбиття кожного атрибуту та вимірює домішки у розділі даних **D** (це набір навчання кортежів із міченими класами) в результаті її розбиття:

$$Gini(\mathbf{D}) = 1 - \sum_{i=1}^{m} p_i^2,$$

де m – кількість класів Ci, $i = 1, \ldots, m$, p_i – ймовірність того, що набір кортежів в **D** відноситься до класу C_i й оцінюється як $|C_{i,D}|/|\mathbf{D}|$; $C_{i,D}$ – множина кортежів

класу C_i в **D**, |**D**| і $|C_{i,D}|$ – кількість кортежів у **D** і $C_{i,D}$, відповідно. Розрізняють такі випадки:

1. **Атрибут A має дискретні значення** $\{a_1, ..., a_v\}$ з **D**. Для визначення найкращого двійкового розбиття на основі A розглянемо всі можливі підмножини, які можна сформувати з використанням відомих значень A. Кожну підмножину S_A можна розглядати як бінарний тест для атрибутів в формі «$A \in S_A$?». З огляду на кортеж, цей тест виконується, якщо значення мітки класу кортежу є одним зі значень, перерахованих в S_A.

Якщо є *v* можливих значень, то існує 2v можливих підмножин. Наприклад, якщо атрибут **дохід** має три можливих значення, а саме: {low, medium, high}, то множина можливих підмножин має вигляд: {{low, medium, high}, {low, medium}, {low, high}, {medium, high}, {low}, {medium}, {high}, {}}. Ми виключаємо з розгляду множину {low, medium, high} і порожню множину, оскільки, концептуально, вони не формують собою підрозділи. Таким чином, існує (2v–2) можливих шляхів формування двох розділів даних з бази D, заснованих на двійковому розбитті на основі A.

При розгляді бінарного розбиття обчислюють зважену суму домішок в результаті кожного розбиття. Наприклад, якщо двійкове розбиття на основі атрибуту A формує підрозділи D_1 і D_2 з D, то $Gini_A(D)$ для заданого атрибуту кожного з можливих варіантів двійкового розбиття розрахують за формулою:

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2).$$

для дискретно-значного атрибуту підмножина, якій відповідає мінімальне значення індексу Джині для певного атрибута, вибирається в якості варіанту розбиття на підмножини.

2. **Атрибут A має неперервні значення**. Для неперервних значень атрибутів потрібно розглянути кожне можливе значення точки розбиття split_point (в якості можливого значення split_point береться середина між кожною парою відсортованих сусідніх значень). Точку, яка має мінімальний індекс Джині для заданого атрибуту, беруть як **split_poin** цього атрибута. Точка

split_poin реалізує розбиття, якщо D1 – це підмножина кортежів з D, які відповідають умові A ≤ split_point, D2 – це підмножина кортежів з D, які відповідають умові A > split_point.

Критерій розбиття. Зниження домішок, які можуть виникнути на основі бінарного розбиття на дискретному або неперервному значенні атрибуту A визначають за формулою: $\Delta Gini(A) = Gini(D) - Gini_A(D)$. Атрибут, який максимізує скорочення домішки (що еквівалентно твердженню «має мінімальний індекс Джині»), обирають за точку розбиття атрибута. Цей атрибут і підмножини його розбиття (для дискретних значень атрибуту) або *split_poin* (для неперервних значень) разом утворюють **критерій розбиття**.

17. РБФ–мережа, сформована на основі Дерева класифікації

Мета роботи: ознайомитися з 1) принципами функціонування та навчання РБФ–мережі, сформованої на основі Дерева класифікації; 2) методом керування кількістю прихованих нейронів та визначення їх параметрів на основі Дерева класифікації. *Об'єкт дослідження*: РБФ–мережа, сформована на основі Дерева класифікації.

Питання для опрацювання

1. РБФ–мережа, сформована на основі Дерева розв'язків [16].

1. Завдання до практичної роботи

1. Розробити РБФ–мережу, здатну розпізнавати об'єкти, подані у вигляді точок 2-вимірного простору. Мережа сформована на основі Дерева розв'язків.

Постановка задачі

На основі навчальної вибірки сформовано Дерево розв'язків, яке генерує функцію c: $\Re^n \to$L, де L – множина міток класів. Для бінарного класифікатора маємо L={0, 1}. Якщо на вхід цього Дерева подається вектор $\mathbf{x}=\mathbf{x}^i=[x_1...x_n]^T$, i=1,..., Q, то його вихід $c(\mathbf{x})$ дорівнюватиме значенню «1», якщо мітка класу більше або дорівнює значенню «0,5», та «0», якщо мітка класу менше «0,5».

Нехай задано навчальну вибірку у вигляді пар даних вхід–ціль: $\{\mathbf{x}^1, t^1\}$, …, $\{\mathbf{x}^Q, t^Q\}$, яка генерується функцією $t^i = f(\mathbf{x}^i)$, i=1, …, Q, де $\mathbf{x}^i = [x_1^i \ \dots \ x_n^i]^T$ – вектор вхідних даних з елементами $x_j^i \in \Re$; t^i – бажаний відгук. Функція $f(\cdot)$ вважається невідомою, але задано множину її реалізацій: $T = \left\{(x_1^i, ..., x_n^i, t^i),\ i = 1, 2, ..., Q, n > 1\right\}$.

Побудувати РБФ-мережу для визначення функції $F(\mathbf{w}, \mathbf{x}^i)$, яка апроксимує функцію $f(\mathbf{x})$, описуючи перетворення вхідного сигналу у вихідний, і задовольняє умову $\frac{1}{Q}\sum_{i=1}^{Q}\left|F(\mathbf{w},\mathbf{x}^i) - t^i\right| < \varepsilon,$ де ε – деяке додатне число, яке називають нев'язкою. Ініціалізацію РБФ-мережі (функції $F(\mathbf{w}, \mathbf{x}^i)$) виконано на основі Дерева розв'язків. Вагові коефіцієнти вихідного шару визначаємо на основі псевдообернего правила.

Порядок виконання роботи

Крок 1. За допомогою Дерева розв'язків розбити простір прикладів на майже однорідні області у вигляді прямокутників. Побудувати відповідний графік.

Крок 2. Пов'язати кожний прямокутник із одним РБФ-нейроном, його параметри (центр, дисперсія) залежать від розташування і вимірів прямокутника.

Крок 3. Організувати нейрони в один прихований шар. Визначити вагу вихідного шару.

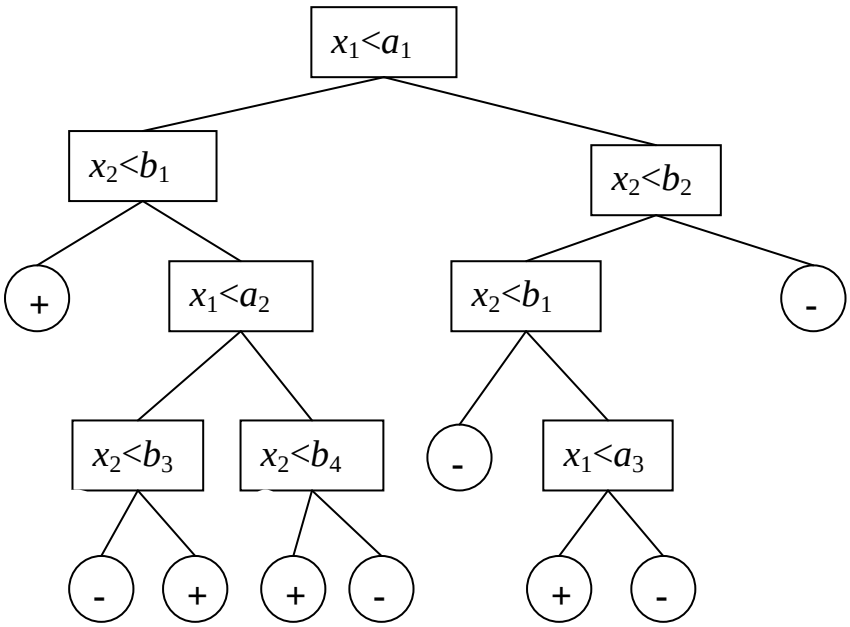
Крок 4. Перевірити РБФ-мережу на здатність розв'язувати задачу класифікації на множині навчання та тестування.

2. Контрольні запитання та завдання

1. Опишіть структуру РБФ-мережі з одним вихідним нейроном.

2. Опишіть алгоритм формування РБФ-мережі, сформованої на основі Дерева розв'язків.

Задачі для самостійного розв'язання



№ п.п.	Умова задачі
1	$a\in[1, 11]$; $b\in[1, 11]$: a_1=6, a_2=3, a_3=9; b_1=4, b_2=8, b_3=7, b_4=9
2	$a\in[5, 15]$; $b\in[5, 15]$: a_1=10, a_2=7, a_3=13; b_1=8, b_2=12, b_3=11, b_4=13
3	$a\in[-1, 9]$; $b\in[-1, 9]$: a_1=4, a_2=1, a_3=7; b_1=2, b_2=6, b_3=5, b_4=7
4	$a\in[10, 20]$; $b\in[10, 20]$: a_1=15, a_2=12, a_3=18; b_1=13, b_2=17, b_3=16, b_4=18
5	$a\in[-5, 5]$; $b\in[-5, 5]$: a_1=0, a_2=-3, a_3=3; b_1=-2, b_2=2, b_3=1, b_4=3
6	$a\in[15, 25]$; $b\in[15, 25]$: a_1=20, a_2=17, a_3=23; b_1=18, b_2=22, b_3=21, b_4=23
7	$a\in[-10, 0]$; $b\in[-10, 0]$: a_1=-5, a_2=-8, a_3=-2; b_1=-7, b_2=-3, b_3=-4, b_4=-2
8	$a\in[20, 30]$; $b\in[20, 30]$: a_1=25, a_2=22, a_3=28; b_1=23, b_2=27, b_3=26, b_4=28
9	$a\in[-15, -5]$; $b\in[-15, -5]$: a_1=-10, a_2=-13, a_3=-7; b_1=-12, b_2=-8, b_3=-9, b_4=-7
10	$a\in[25, 35]$; $b\in[25, 35]$: a_1=30, a_2=27, a_3=33; b_1=28, b_2=32, b_3=31, b_4=33
11	$a\in[-20,-10]$; $b\in[-20,-10]$: a_1=-15, a_2=-18, a_3=-12; b_1=-17, b_2=-13, b_3=-14, b_4=-12
12	$a\in[2, 12]$; $b\in[2, 12]$: a_1=7, a_2=4, a_3=10; b_1=5, b_2=9, b_3=8, b_4=10
13	$a\in[-2, 8]$; $b\in[-2, 8]$: a_1=3, a_2=0, a_3=6; b_1=1, b_2=5, b_3=4, b_4=6
14	$a\in[3, 13]$; $b\in[3, 13]$: a_1=8, a_2=5, a_3=11; b_1=6, b_2=10, b_3=9, b_4=11

15	$a\in[-3, 7]$; $b\in[-3, 7]$: a_1=2, a_2=-1, a_3=5; b_1=0, b_2=4, b_3=3, b_4=5
16	$a\in[4, 14]$; $b\in[4, 14]$: a_1=9, a_2=6, a_3=12; b_1=7, b_2=11, b_3=10, b_4=12
17	$a\in[-4, 6]$; $b\in[-4, 6]$: a_1=1, a_2=-2, a_3=4; b_1=-1, b_2=3, b_3=2, b_4=4
18	$a\in[1, 9]$; $b\in[1, 9]$: a_1=5, a_2=2, a_3=8; b_1=3, b_2=7, b_3=6, b_4=8
19	$a\in[1, 10]$; $b\in[1, 10]$: a_1=6, a_2=1,5, a_3=7; b_1=2, b_2=6, b_3=5, b_4=7
20	$a\in[0, 8]$; $b\in[0, 9]$: a_1=6, a_2=1,5, a_3=7; b_1=2, b_2=6, b_3=5, b_4=7
21	$a\in[0, 11]$; $b\in[0, 11]$: a_1=4, a_2=3, a_3=9; b_1=2, b_2=4, b_3=6, b_4=9

Аудиторна робота

Приклад 17.1.

1. Спочатку переконаємося, що отримана в результаті навчання на основі даного методу РБФ-мережа здатна моделювати звичайне Дерево розв'язків (рис. 17.2). Для цього змоделюємо РБФ-мережу, яка відповідає цьому Дереву класифікації для точок із області X=[0, 10]×[0, 10].

2. Визначимо множину навчання S_1, куди входять точки (вектори) з усіх дев'яти підобластей, та множину тестування S_2:

S_1={[1 1]^T, [1 8]^T, [1 5]^T, [3,5 5,5]^T, [4 9]^T, [8 2]^T, [6,5 5]^T, [8,5 5]^T, [8 8]^T};
значення класів C={1; 1; 0; 1; 0; 0; 1; 0; 0};

S_2={[0,1 0,1]^T, [0,5 8]^T, [0,1 5]^T, [3 5]^T, [2,5 8,5]^T};
значення класів C={1; 1; 0; 1; 0}.

3. Визначимо параметри функції Гаусса (рис. 17.4).

4. Визначимо матрицю інтерполяції **L** та вагу **w**.

Дані навчання

```
fID = fopen('data1.txt','r');
data = fscanf(fID,'%f, %f, %f', [3 inf])';
x=data(:,[1:2]); %input base !count of attributes must be same!
cl=data(:,3); fclose(fID)

function func=make_func(c,sigma,x)
  func=1;  m=size(c);
  for i=1:m(2)
   func=func*exp(-((x(i)-c(i))^2)/(2*sigma(i)^2));
  end
end
function y=calc_group(W,centers,sigma,x)
  y=0; wn=size(W);
  for k=1:wn(1)
      y=y+W(k)*make_func(centers(k,:),sigma(k,:),x)
end; end
```

Етап навчання

```
clear all;close all;clc;
% x=[[1, 1]; [1, 8]; [1, 5]; [3.5, 5.5]; [4, 9]; [8, 2]; [6.5, 5];
[8.5, 5]; [8, 8]]
% T=[1; 1; 0; 1; 0; 0; 1; 0; 0]
fID = fopen('data1.txt','r');
```

```
Data = fscanf(fID,'%f, %f, %f', [3 inf])';
x=Data(:,[1:2]); %input base !count of attributes must be same!
T=Data(:,3); fclose(fID); xn=size(x);

centers=[[0, 0];[0, 4.5];[0, 10];[3.5, 5.5];[3.5, 10];[10, 0];
[6.5, 5]; [10, 5];[10, 10]]
sigma=[[5, 3]; [2,1.5];[2,4];[1.5,2.5];[1.5,2]; [5,3];[1.5,2];
[2,2];[5,3]]
cn=size(centers); G=zeros(xn(1), cn(1));
for i=1:xn(1)
   for j=1:cn(1)
     G(i,j)=make_func(centers(j,:),sigma(j,:),x(i,:));
   end
end
Gplus=inv(G'*G)*G';W=Gplus*T;
spivpad=0;nespivp=0;
Y=zeros(xn(1),1);
for i=1:xn(1)
  tmp=calc_group(W,centers,sigma,x(i,:))
  if tmp>0.5
   Y(i)=1;
  else
   Y(i)=0
  end;
  if Y(i)==T(i)
     spivpad=spivpad+1;
  else
     nespivp=nespivp+1;
  end
end; error1=nespivp
```

Результат `error1=0;`

Матриця ваги:

```
W =   1.1596
     -0.5611
      1.6239
      0.9570
     -0.1996
     -0.1556
      0.9908
     -0.2804
     -0.0504
```

Етап тестування

```
clear all;close all;clc;
% x={[0,1 0,1], [0,5 8], [0,1 5], [3 5], [2,5 8,5]}
% T={1; 1; 0; 1; 0]
fID = fopen('test.txt','r');
test = fscanf(fID,'%f, %f, %f', [3 inf])';
x=test(:,[1:2]); %input base !count of attributes must be same!
T=test(:,3); fclose(fID);
xn=size(x);
```

```
centers=[[0, 0];[0, 4.5];[0, 10];[3.5, 5.5];[3.5, 10]; [10, 0];
[6.5, 5]; [10, 5];[10, 10]]
sigma=[[5, 3];
[2,1.5];[2,4];[1.5,2.5];[1.5,2];[5,3];[1.5,2];[2,2];[5,3]]
W =[ 1.1596, -0.5611,1.6239,0.9570, -0.1996,-0.1556,0.9908,
-0.2804, -0.0504]'
spivpad=0;nespivp=0;Y=zeros(xn(1),1);
for i=1:xn(1)
  tmp=calc_group(W,centers,sigma,x(i,:))
  if tmp>0.5
   Y(i)=1;
  else
   Y(i)=0;
  end;
  if Y(i)==T(i)
     spivpad=spivpad+1;
  else
     nespivp=nespivp+1;
  end
end
error2=nespivp
```

Результат: error2=0

Приклад 17.2: python

```
import numpy as np
from numpy.linalg import inv
import math

def print_list(lis):
 for i in lis:
   print(i)

def fii(c,sigma,x):
 m=len(c); p=1;
 for i in range(m):
  p=p*math.exp(-((x[i]-c[i])**2)/(sigma[i]**2))
 return p

c=[] # c.dat містить центри функцій Гауса
with open("c.dat") as f:
  for i in f:
    c.append([float(j) for j in i.split(", ")])

sigma=[] # sigma.dat містить дисперсії функцій Гауса
with open("sigma.dat") as f:
  for i in f:
    sigma.append([float(j) for j in i.split(", ")])

data=[] # data.dat містить множину точок навчання
with open("data.dat") as f:
  for i in f:
```

```
    data.append([float(j) for j in i.split(", ")])

test=[] # test.dat містить множину точок тестування
with open("test.dat") as f:
  for i in f:
    test.append([float(j) for j in i.split(", ")])

x=[]; y=[]
for i in data:
 x.append([i[0], i[1]]); y.append(i[2])
f=[]
for i in range(len(y)):
 f.append([])
 for j in range(len(c)):
  f[i].append(fii(c[j],sigma[j],x[i]))

ft=np.array(f); f=np.array(f)
ft=ft.transpose()
fa=np.dot(ft,f)
fa=inv(fa)
w=np.dot(np.dot(fa,ft),y)

y1=[]
for i in range(len(y)):
  y1.append(round(sum([F[i][j]*w[j] for j in range(len(c))]),3))

print("На множині навчання отримали такі значення класів: \n", y1)
x=[]; y=[]
for i in test:
 x.append([i[0], i[1]]); y.append(i[2])
f=[]
for i in range(len(y)):
 f.append([])
 for j in range(len(c)):
  f[i].append(fii(c[j],sigma[j],x[i]))

y1=[]
for i in range(len(y)):
  y1.append(round(sum([F[i][j]*w[j] for j in range(len(c))]),3))
print("На множині тестування отримали такі значення класів: \n",
[str(i) for i in y1])
err=sum([abs(y[i]-y1[i]) for i in range(len(y))])
print("Похибка: ", err)
```

Результат: error2=0: y=[1; 1; 0; 1; 0].

```
>>>
========== RESTART: C:\Users\Ирина\Desktop\lr_15RBF_tree\lr_15.py ==
На множині навчання отримали такі значення класів:
 [1.0, 1.0, -0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0]
На множині тестування отримали такі значення класів:
 ['1.158', '1.191', '-0.084', '0.858', '0.364']
Похибка:  0.939
```

Теоретичний матеріал

1. РБФ-мережа, сформована на основі Дерева розв'язків [16]

Розглянемо, як можна використати Дерево розв'язків при формуванні РБФ-мережі. РБФ-мережу зображено на рис. 17.1.

Тут функція активації вихідних нейронів є лінійною, функція активації i-го нейрона прихованого шару має такий вигляд

$$\varphi_i(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}-\boldsymbol{\mu}_i\|^2}{2\sigma_i^2}\right),$$ де

$\mu_i=[\mu_{i1},\ldots,\ \mu_{in}]$ – центр, а σ_i^2 – дисперсія i-ого РБФ–нейрона.

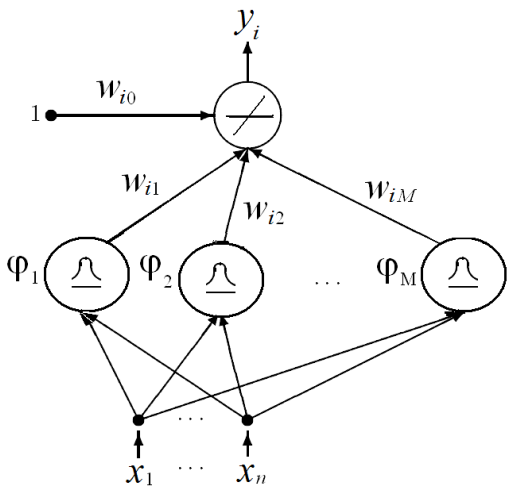


Рис. 17.1. РБФ-мережа з одним вихідним нейроном

Щоб вирішити, яку мітку класу необхідно призначити входу **x**, мережа спочатку відображає **x** у M-вимірний вектор $\boldsymbol{\varphi}(\mathbf{x})=[\varphi_1(\mathbf{x})\ \ldots\ \varphi_M(\mathbf{x})]^T$.

Значення, отримане на виході i-го вихідного нейрона, обчислюють як $y_i = \sum_{j=0}^{M} w_{ij}\varphi_j(\mathbf{x}),$ $i=1,\ \ldots,\ m$; де w_{ij} – ваговий коефіцієнт зв'язку між j-им прихованим нейроном та i-им вихідним нейроном (w_{i0} з'єднується із фіксованим $\varphi_0=1$). Зазвичай приклад з множини навчання (або тестування) помічається i-им класом, якщо він визначається як: $y_i = \max_k\{y_1,\ \ldots,\ y_m\}$.

Поведінка РБФ-мережі залежить від таких параметрів: кількості нейронів прихованого шару, значень параметрів функцій Гауса (центрів і дисперсій) і значень вагових коефіцієнтів вихідного шару. Метод визначення вагових

коефіцієнтів вихідного шару відносно простий – можна навчити мережу, наприклад, на основі псевдооберненої матриці [1].

Нехай $\boldsymbol{\mu}_{ik}$ – k-а координата центра μ_i, σ_{ik}^2 – дисперсія i-ої функції Гауса уздовж k-го атрибута. Тоді для обчислення виходу i-го нейрона прихованого шару в областях із n атрибутами функцію (1) можна записати у вигляді:

$$\varphi_i(\mathbf{x}) = \prod_{k=1}^{n} \exp\left(-\frac{\|x_k - \mu_{ik}\|^2}{2\sigma_{ik}^2} \right).$$

Механізм визначення функції Гауса за допомогою гіперпаралелепіпедів, отриманих на основі генератора Дерева розв'язків. Для розв'язання зазначеної вище проблеми дослідження використовується метод, який базується на ідеї зв'язування кожного нейрону з деякою відносно однорідною областю простору. Щоб створити однорідні області у формі гіперпаралелепіпедів, метод використовує генератор Дерева розв'язків. Опишемо метод, який вирішує задачу визначення параметрів РБФ-мереж.

Для визначення майже однорідних областей існують методи побудови індукційних Дерев розв'язків. Покажемо, що Дерево розв'язків на основі тестування одного атрибута визначає набір однорідних гіперпаралелепіпедів, які можна перетворити у РБФ-мережу. Приклад Дерева розв'язків (рис. 17.2) та відповідний йому двовимірний простір $\Re^2$ атрибутів досліджуємих об'єктів, який розбивають на області у вигляді прямокутників, зображено на рис. 17.3. Кожна гілка Дерева складається з множини бінарних тестів одного атрибуту і закінчується листом, який містить мітку класу (розглядаються тільки класи: C_1=«+», C_2=«–»).

Щоб визначити клас, до якого належить вектор атрибутів **x**, класифікатор перевіряє **x** цим тестом, починаючи з кореня. Результат кожного тесту вирішує, чи продовжувати аналіз з лівої чи з правої гілки. Коли **x** досягає листа, дерево призначає йому мітку класу, пов'язану з цим листом. Для цього прикладу квадрат зі сторонами (0, 10) розбивається на прямокутники (для n атрибутів

будемо мати гіперпаралелепіпеди), кожний з яких містить точку, що зображує нейрон (рис. 17.4) і помічає розташування центру функції Гауса.

Вихід нейрона є максимальним для векторів, розташованих в околі точки центру. Його значення буде зменшуватися зі зростанням відстані до цієї точки. На межі прямокутника вихід дорівнює деякому наперед встановленому значенню *a*, однаковому для всіх нейронів. Деякі прямокутники знаходяться в оточенні (усередині) інших (рис. 17.3).

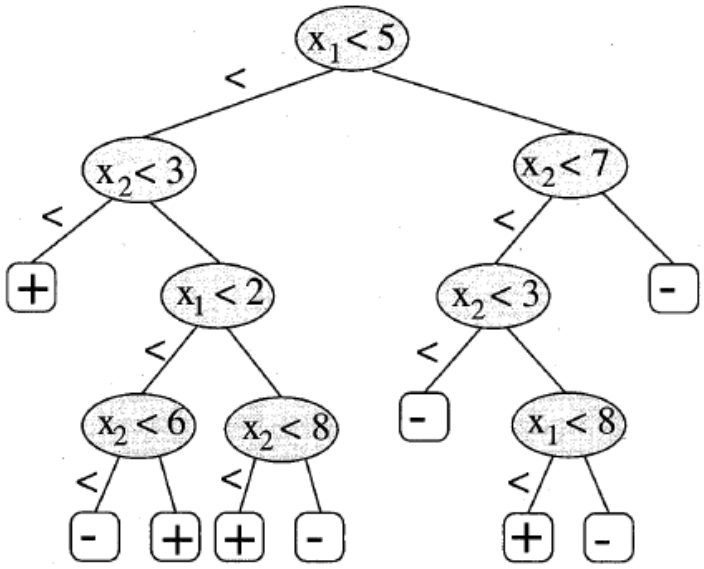


Рис. 17.2. Дерево розв’язків, яке розбиває двовимірний простір на прямокутники

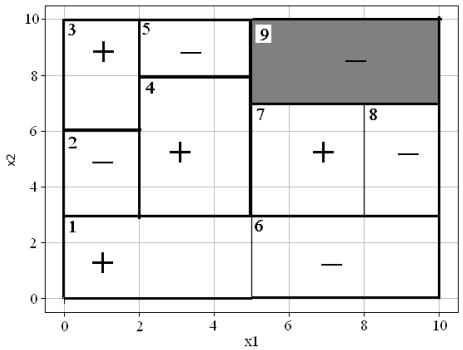


Рис. 17.3. Прямокутники, які відповідають Дереву розв’язків з рис. 17.2

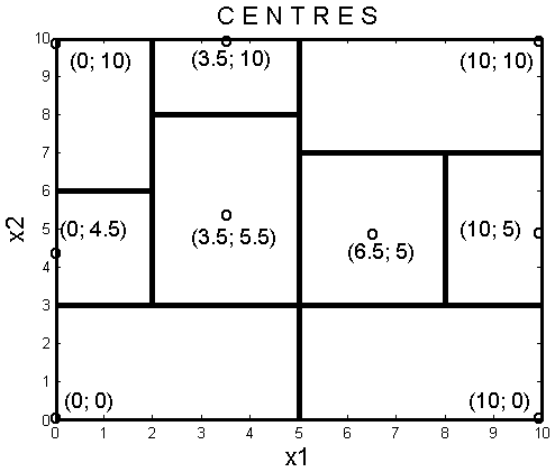


Рис. 17.4. Розташування центрів функції Гауса

У таких прямокутниках точки центру розміщено в геометричних центрах, а для інших прямокутників, розташованих «на межі», застосовується інший підхід.

Значення функцій Гауса, які відповідають атрибуту x_2 з рис. 17.3, зображено на рис. 17.5 (точки *max* і *min* є максимальними і мінімальними значеннями атрибуту x_2, які спостерігаються на множині навчання). Залежно від того, чи розташована область усередині простору, ця максимальна відстань знаходиться або в геометричному центрі прямокутника, або на його межі.

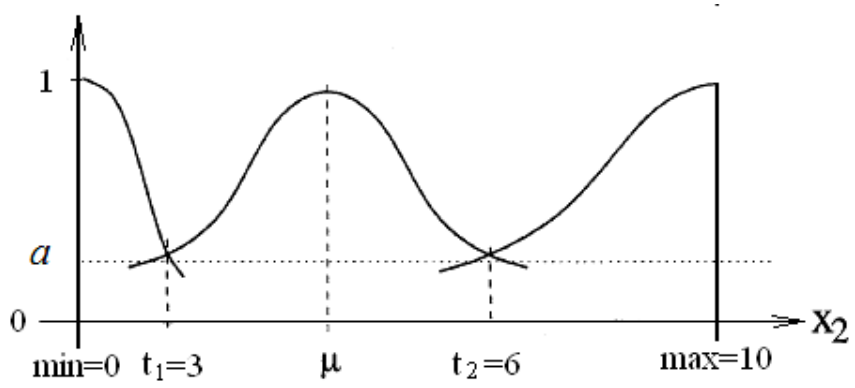


Рис. 17.5. Функції Гаусса, які відповідають атрибуту x_2 з рис. 17.3

Опишемо алгоритм, який дозволяє розбити простір прикладів на відносно однорідні області у вигляді гіперпаралелепіпедів, кожний з яких асоційовано з одним із РБФ-нейронів:

Крок 1. Визначення координат центрів функцій Гауса μ. Для кожного гіперпаралелепіпеда, визначеного на основі Дерева розв'язків, встановлюємо місце розташування центру функції Гауса μ. «Межа простору» визначається як максимальне або мінімальне значення заданого атрибуту у множині навчання.

Крок 2. Визначення дисперсії. Розміри гіперпаралелепіпедів окремо для кожного атрибута визначають дисперсію функції Гауса. Наприклад, гілка Дерева розв'язків, яка відповідає умові $x_1 \geq 5$ і $x_2 \geq 7$ (рис. 17.2), описує прямокутник за номером «9» сірого кольору, визначений як $x_1 \in [5, 10]$ і $x_2 \in [7, 10]$. Можна побачити, що $\sigma_{91}=5$ і $\sigma_{92}=3$. Функція активації дев'ятого РБФ-нейрона буде визначена за формулою:

$$\varphi_9(\mathbf{x}) = \exp\left(-\frac{(x_1-10)^2}{25}\right)\cdot\exp\left(-\frac{(x_2-10)^2}{9}\right).$$

Крок 3. *Визначення ваги вихідного шару.* Нехай трансформовані на основі РБФ φ–приклади множини навчання зафіксовані у вигляді матриці **X** так, що кожний рядок зображує один приклад, а *i*-ий стовпчик містить значення φ_i цього прикладу. Нульовий атрибут $\varphi_0=0$.

Нехай **C** є матрицею класифікації, де кожний стовпчик підтримує одну мітку класу: якщо *r*-ий приклад маркується *j*-им класом, то *j*-е поле в *r*-ому рядку **C** дорівнює значенню «1», а всі інші поля в цьому рядку містять «–1» (або «0»). Завдання полягає в тому, щоб визначити вектор ваги **W**, який дозволить мінімізувати середньоквадратичну похибку вигляду (**XW** – **C**). Цього можна досягти за допомогою псевдооберненої матриці $\mathbf{X}^+$, користуючись тим фактом, що середньоарифметична похибка мінімізована, якщо:

$$\mathbf{W}=(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{C}=\mathbf{X}^+\mathbf{C}. \qquad (17.1)$$

Визначимо *алгоритм проектування РБФ-мережі на основі генерації Дерева розв'язків.*

Крок 1. Розбити простір прикладів на майже однорідні області у вигляді гіперпаралелепіпедів (це можна реалізувати за допомогою генератора Дерева розв'язків).

Крок 2. Пов'язати кожний гіперпаралелепіпед із одним РБФ-нейроном (його параметри залежать від розташування і вимірів гіперпаралелепіпеда).

Крок 3. Організувати нейрони в один прихований шар. Визначити вагу вихідного шару. Вагу вихідного шару визначають за формулою (17.1), яка максимізує точність класифікації мережі на множині навчання.

18-19. Нечітка кластеризація даних. Алгоритм нечіткої кластеризації даних FCM

Мета роботи: ознайомитися із можливою реалізацією методу нечіткої кластеризації даних (алгоритм FCM); сформувати вміння розв'язувати задачу кластерного аналізу на основі алгоритму FCM. *Об'єкт дослідження* – алгоритм нечіткої кластеризації даних FCM.

Питання для опрацювання

1. Загальна характеристика задач кластерного аналізу [3].
2. Задача нечіткої кластеризації та алгоритм її розв'язання [3].
3. Засоби розв'язання задачі нечіткої кластеризації в пакеті Fuzzy Logic Toolbox системи MatLab [3].
4. Приклад програмної реалізації алгоритму FCM [3]

1. Задання до практичної роботи

1.Вивчити теоретичний матеріал.

2. Підготувати відповідні дані до задачі діагностики захворювань у вигляді файлу «*.*txt*».

3. Використовуючи підготовлені дані, зафіксовані у текстовому файлі, написати програму (m-файл), яка реалізує алгоритм нечіткої кластеризації даних FCM. Продемонструвати та пояснити роботу алгоритму (програм на мовах С та MatLab) на прикладах. Візуалізувати результати обчислень в системі MatLab (побудувати графіки, що інтерпретують одержаний результат).

4. Порівняти отриманий результат з результатом роботи внутрішньої функції системи MatLab – FCM.

5. Написати програму на мові Python, яка реалізує алгоритм нечіткої кластеризації даних FCM.

2. Контрольні запитання та завдання

1. Що таке нечіткі відношення? Які способи їх зображення ви знаєте?
2. Назвіть основні характеристики нечітких відношень.
3. Назвіть основні властивості бінарних нечітких відношень, визначених на одному універсумі.
4. Опишіть алгоритм розв'язання задачі нечіткої кластеризації методом нечітких с-середніх. Визначіть відстань між кластерами.

Задачі для самостійного розв'язання: алгоритм нечіткої кластеризації даних FCM (комп'ютерний практикум № 7)

Аудиторна робота

Приклад 18.1.

1. Підготовка даних. Необхідно проаналізувати записи з бази даних, в якій міститься інформація про наявність (відсутність) діабету у жителів Феніксу. Кожен рядок має 9 значень атрибутів, а саме:

1. Скількі разів була вагітною.

2. Концентрація глюкози в плазмі через дві години після навантаження – тест на толерантність до глюкози

3. Діастолічний артеріальний тиск (мм рт.ст.)

4. Трицепс (товщина шкірних складок): мм

5. Рівень інсуліну в сиворотці крові через дві години

6. Індекс маси тіла – вага (кг / (зріст в м)^2)

7. Функція діабет за спадковістю

8. Вік: років

9. *Діагноз*: бінарна змінна клас: {0, 1}, клас 1 означає «у хворого є діабет».

Оскільки маємо два необхідних кластери, то будемо використовувати кластеризацію на два кластери. Всі дані бази були перенесені в файл *lab2.txt* (Додаток В).

Розробка програми розв'язання задачі кластеризації

Відстань Евкліда. Формула традиційної відстані між двома точками $\mathbf{p}=[p_1 \ldots p_n]^T$ та $\mathbf{q}=[q_1 \ldots q_n]^T$ для простору Евкліда має вигляд:

$$\sqrt{(p_1-q_1)^2+\ldots+(p_n-q_n)^2}=\sqrt{\sum_{i=1}^{n}(p_i-q_i)^2}$$

Відстань Мінковського порядку m між двома точками $\mathbf{p} = [p_1 \ldots p_n]^T$, $\mathbf{q}=[q_1 \ldots q_n]^T \in \Re^n$ визначається таким чином: $\left(\sum_{i=1}^{n}|p_i-q_i|^m\right)^{1/m}$. Якщо $m=2$ маємо відстань Евкліда, якщо $m=1$, то Манхетенську відстань, якщо $m\to\infty$, то відстань Чебишева $\lim_{p\to\infty}\left(\sum_{i=1}^{n}|p_i-q_i|^m\right)^{1/m}=\max_{i=1,n}|p_i-q_i|$.

Відстань Хеммінга – кількість позицій, у яких відповідні цифри двох двійкових слів однакової довжини різні.

Код в середовищі Matlab

Дані

```
6,148,72,35,0,33.6,0.627,50,1
1,85,66,29,0,26.6,0.351,31,0
8,183,64,0,0,23.3,0.672,32,1 …
```

А. Із використанням вбудованих функцій

```
Clc; clear all; close all;
f=fopen('lab6.txt','r');
[X,count]=fscanf (f,'%f,%f,%f,%f,%f,%f,%f,%f,%f',[9 inf]);
```

```
fclose(f);
X=X';X=X(1:24,1:9);
[centers,U]=fcm(X,2); maxU = max(U);
index1 = find(U(1,:) == maxU);
index2 = find(U(2,:) == maxU);
str1=struct('X', [X(index1,1),X(index1,2),X(index1,3)], 'U',
U(1:2,index1), 'V', V(1,1:3));
str2=struct('X', [X(index2,1),X(index2,2),X(index2,3)], 'U',
U(1:2,index2), 'V', V(2,1:3));
figure(1)
plot3(str1.X(:,1), str1.X(:,2), str1.X(:,3), 'xk', str2.X(:,1),
str2.X(:,2), str2.X(:,3),'ok');hold on;
plot3(str1.V(1,1),str1.V(1,2),str1.V(1,3), '*k',V(2,1), V(2,2),
V(2,3), '*k');
legend('Кластер 1','Кластер 2', 'Location','S');
grid on;str1.U';str2.U';
a=zeros(20,1); a(index2)=1;
MSE=mean((a-cl).^2) % =sum(abs(a-cl))/20
```

Б. Без використання вбудованих функцій

Нехай маємо множину даних $X=\{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{n-1}\}$ – набір із n векторів в R^k. Алгоритм FCM реалізує класифікацію, мінімізуючи цільову функцію (функцію похибки): $J(U,V)=\sum_{i=0}^{n-1}\sum_{j=0}^{c-1}\left(\mu_{ij}\right)^m \left\|\mathbf{x}_i-\mathbf{v}_j\right\|^2$, де μ_{ij} – ступінь належності $\mathbf{x}_i$ до нечіткого кластеру з центром $\mathbf{v}_j$, який задовольняє умову $\sum_{j=0}^{c-1}\mu_{ij}=1$, i=0, ..., n – 1, $\left\|\mathbf{x}_i-\mathbf{v}_j\right\|$ – відстань Евкліда між точками $\mathbf{x}_i$ та $\mathbf{v}_j$. Немає жодного теоретичного обгрунтування для оптимального вибору m, але зазвичай m=2. Матриця $\mathbf{U}=\{\mu_{ij}\}$ – нечітка матриця розбиття розміру $n\times c$ і $V=\{\mathbf{v}_0, \mathbf{v}_1, ..., \mathbf{v}_{c-1}\}$ – множина центрів кластерів ($\mathbf{v}_j\in R^k$). **Алгоритм FCM**:

Крок 1. Ініціалізувати матрицю $\mathbf{U}=\{\mu_{ij}\}$ із випадковими значеннями таким чином, щоб: $\mu_{ij}\in[0, 1]$, i = 0, ..., n – 1; j = 0,..., c – 1; $\sum_{j=0}^{c-1}\mu_{ij}=1$, i = 0, ..., n – 1.

Крок 2. Обчислити нечіткі центри $\mathbf{v}_j$ за формулою: $v_{jl}=\frac{\sum_{i=0}^{n-1}\left(\mu_{ij}\right)^m x_{il}}{\sum_{i=0}^{n-1}\left(\mu_{ij}\right)^m}$, j = 0, ..., c–1; l = 0, ..., k – 1.

Крок 3. Обчислити відстань $d_{ij}=\left\|\mathbf{x}_i-\mathbf{v}_j\right\|$, i = 0, ..., n – 1; j = 0, ..., c – 1;

Крок 4. Відновити матрицю **U**={μ_{ij}} за правилом: якщо $d_{ij} \neq 0$, то $\mu_{ij} = \dfrac{1}{\sum_{l=0}^{c-1}\left(d_{ij}^2 / d_{il}^2\right)^{1/(m-1)}}$, інакше $\mu_{ij} = 1$.

Крок 5. Повторювати кроки 2, 3, 4 доки значення *J* не досягне мінімуму або не буде виконано наперед задану кількість ітерацій (або зменшення *J* щодо попередньої ітерації дуже мале).

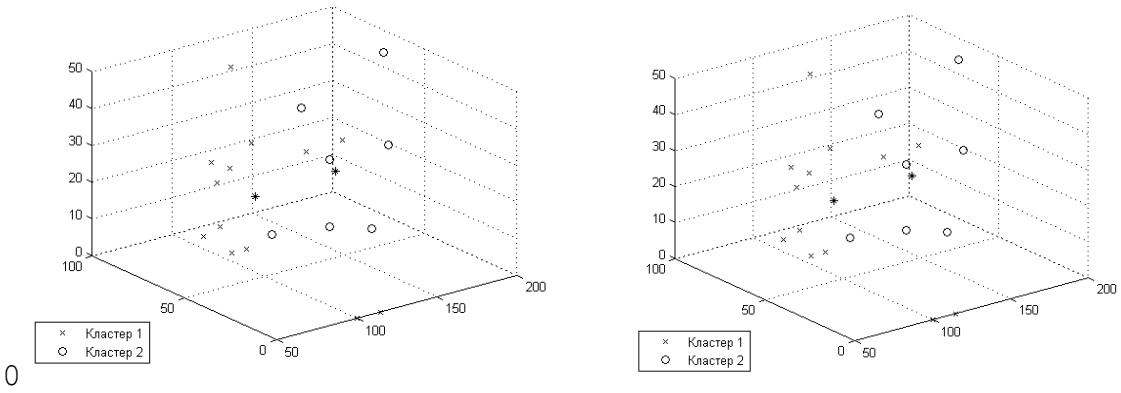
```
clc; close all; clear all;
f=fopen('lab6.txt','r');
[X,count]=fscanf (f,'%f,%f,%f, %f, %f, %f, %f, %f, %f', [9 inf]);
fclose(f);

X=X';cl=zeros(1:20,1);cl=X(1:20,9); X=X(1:20,2:4);
r=corrcoef(X); N=2; % N — кількість кластерів
[K,L]=size(X); U=rand(N,K); s=sum(U); normator=s;
U=U./normator(ones(N,1),:); % U collumns have the sum of 1
U=U';n=K,k=L,c=N;iter=50; count = 0;
while count < iter
  for j = 1:c
   for l = 1:k
     t1 = 0.0; t2 = 0.0;
     for i = 1:n
      t1=t1+U(i,j)*U(i,j)*X(i,l); t2=t2+U(i,j)*U(i,j);
     end;
     V(j,l) = t1/t2;
  end; end;
  for i = 1:n
   for j = 1:c
     dist(i,j) = FCM_dist(X(i,:),V(j,:)); % euclid_dist
  end; end
  for i = 1:n
   for j = 1:c
     t3 = 0.0;
     if dist(i,j) == 0
      U(i,j) = 1.0;
     else
      for p = 1:c
        t3 = t3 + (dist(i,j)*dist(i,j))/(dist(i,p)*dist(i,p));
     end; end
     U(i,j) = 1.0 / t3;
   end;  end;
  count = count + 1;
end
U=U'; maxU = max(U);
index1 = find(U(1,:) == maxU);
index2 = find(U(2, :) == maxU);
str1=struct('X', [X(index1,1),X(index1,2),X(index1,3)], 'U',
U(1:2,index1), 'V', V(1,1:3));
```

```
str2=struct('X', [X(index2,1),X(index2,2),X(index2,3)], 'U',
U(1:2,index2), 'V', V(2,1:3));
figure(1)
plot3(str1.X(:,1), str1.X(:,2), str1.X(:,3), 'xk', str2.X(:,1),
str2.X(:,2), str2.X(:,3),'ok'); hold on;
plot3(str1.V(1,1),str1.V(1,2),str1.V(1,3), '*k',V(2,1), V(2,2),
V(2,3), '*k');
legend('Кластер 1','Кластер 2', 'Location','S');
grid on;str1.U';str2.U';
a=zeros(20,1); a(index2)=1; MSE=mean((a-cl).^2)
```

FCM_dist.m

```
function [d] = dist(q,w) % function distance.m
 x=0;
 for i=1:length(w)
     x=x+(q(i)-w(i))^2;
 end;  d=x^0.5;
end
```

Результат



0

а *б*

```
str1.U=0.9234  0.9485  0.5222  0.7870  0.8931  0.6477  0.5529
0.7212    0.6989    0.6771    0.8579    0.8240    0.8822
      0.0766  0.0515  0.4778  0.2130   0.1069  0.3523  0.4471
0.2788    0.3011    0.3229    0.1421    0.1760    0.1178

str2.U=0.2093  0.0907  0.1658  0.0710 0.3553  0.0888  0.0073
       0.7907  0.9093  0.8342  0.9290 0.6447  0.9112  0.9927
```

Теоретичні відомості

1. Загальна характеристика задач кластерного аналізу [3, с. 29–31]. Здатність розпізнавати й класифікувати образи є однією з фундаментальних характеристик інтелекту людини. Вона відіграє ключову роль у сприйнятті навколишнього світу. Як область досліджень теорія розпізнавання образів почала розвиватися разом із появою та розвитком комп’ютерних технологій.

У загальному випадку, розпізнавання образів визначається як процес пошуку структури даних і класифікації даних на категорії (при цьому дані однієї групи пов'язані між собою більше, ніж дані різних груп). Існують три фундаментальні проблеми розпізнавання образів:

1. *Проблема зображення вхідних даних*. У загальному випадку кожний об'єкт можна описати вектором значень його характеристик: $\mathbf{p}=[p_1 \ldots p_q]^T$.

2. *Проблема вибору основних характеристик (атрибутів) об'єкта серед вхідних даних.* Атрибути повинні відображати властивості, притаманні окремим класам образів.

3. *Проблема визначення оптимального розв'язку задачі класифікації даного образу шляхом визначення значення деякої функції розпізнавання кожного класу*. Ця функція визначає номер класу, якому відповідає окремий образ-вектор.

Задача класифікації об'єктів є предметом дослідження кластерного аналізу, який відіграє важливу роль в розпізнаванні образів. Терміном «кластерний аналіз» прийнято позначати сукупність методів і процедур, призначених для формування однорідних класів у довільній проблемній області. Ще в середині 1960–х років було показано доцільність використання основних положень теорії нечітких множин при розв'язанні задач розпізнавання образів і кластерного аналізу.

Необхідність обробки великих обсягів інформації, пов'язаних з розв'язанням неформалізованих і погано формалізованих задач різної фізичної природи, вимагає розвитку нових наукових напрямків, серед яких важливу роль відіграють прикладна статистика та методи аналізу даних. Методологія застосування методів прикладної статистики грунтується на припущенні про ймовірнісну інтерпретацію інформації, яка аналізується, й отриманні закономірностей, які мають стохастичний характер. Методи аналізу даних, складовою частиною яких є методи кластерного аналізу, навпаки, не використовують апріорних припущень про ймовірнісну природу інформації, а

керуються тільки евристичними міркуваннями щодо характеру та особливостей досліджуваної сукупності об’єктів.

Кластерний аналіз включає в себе сукупність методів і алгоритмів, призначених для знаходження деякого розбиття досліджуваної сукупності об’єктів на підмножини подібних об’єктів.

При виявленні кластерів у множині даних повинні виконуватися наступні умови: 1) кожний кластер повинен відображати концептуально однорідну категорію, тобто містити схожі об’єкти; 2) сукупність всіх кластерів повинна бути вичерпною (охоплювати всі об’єкти досліджуваної множини); 3) кластери повинні бути взаємно-виключними (жоден з об’єктів досліджуваної сукупності не повинен одночасно належати двом різним кластерам).

Формально задачею кластерного аналізу заданої множини об’єктів є визначення деякого теоретико-множинного розбиття цієї множини на підмножини, які не перетинаються, так, щоб елементи однієї підмножини відрізнялися між собою значно менше, ніж елементи різних підмножин.

2. Задача нечіткої кластеризації та алгоритм її розв’язання [3, с. 31–37]. Вимогу однозначності кластеризації елементів проблемної області можна послабити за допомогою методів нечіткої кластеризації, розглянувши нечіткі кластери та відповідні їм функції належності.

У загальному випадку завданням нечіткої кластеризації є визначення нечіткого розбиття (або нечіткого покриття) множини елементів, яке утворює структуру нечітких кластерів. Це завдання зводиться до знаходження ступенів належності елементів універсуму шуканим нечітким кластерам, які в сукупності визначають нечітке розбиття множини елементів.

Розглянемо формальну постановку задачі нечіткого кластерного аналізу. Нехай досліджуваною сукупністю даних є скінченна множина об’єктів кластеризації $A=\{a_1, ..., a_n\}$, кожний з яких описується скінченною множиною ознак (атрибутів) $P=\{p_1, ..., p_q\}$, де p_i кількісно зображує деяку властивість (характеристику) елементів проблемної області.

Передбачається, що для кожного з об'єктів кластеризації певним чином вимірюються всі ознаки множини P. Кожному з елементів $a_i \in A$ ставиться у відповідність вектор $\mathbf{x}_i=[x_1^i \ \ldots \ x_q^i]^T$, де x_j^i – числове значення ознаки $p_j \in P$ для об'єкта даних a_i. Для визначеності будемо вважати, що всі x_j^i приймають деякі дійсні значення.

Процес вимірювання властивостей об'єктів даних може бути реалізований у різних шкалах, кожна з яких характеризується деяким перетворенням даних. У зв'язку з цим, розрізняють такі типи шкал:

Шкала найменування (або номінальна шкала) є найпростішою з усіх шкал виміру, оскільки може бути використана для встановлення відношення еквівалентності елементів відносно визначеної ознаки. В процесі вимірювання кожній властивості об'єкта ставиться у відповідність деякий символ. Перетворенням в шкалах найменувань є взаємно-однозначне відображення між двома множинами значень ознак об'єкта. Прикладом шкали найменувань є бінарна шкала, яка складається з двох елементів, що позначаються довільними символами, наприклад {0, 1} або {+, –}. До ознак, які вимірюються в шкалах найменувань, можна віднести стать людини, марку автомобіля, назву вулиць і міст.

Порядкова шкала. В процесі вимірювання ознаці об'єкта ставиться у відповідність, як правило, деяке натуральне або ціле число. Перетворенням у шкалах порядку є довільне монотонно зростаюче відображення між двома множинами значень ознак. До прикладів ознак, які вимірюються в порядкових шкалах, належать бали або оцінки на іспитах.

Шкала інтервалів. В процесі вимірювання ознаці об'єкта ставиться у відповідність, як правило, деяке дійсне число. Перетворенням в шкалах інтервалів є довільне лінійно зростаюче відображення між двома множинами значень ознак. Характерною властивістю цієї шкали є відсутність абсолютного нуля. До прикладів ознак, які вимірюються в інтервальній шкалі, належать температури за Цельсієм і Фаренгейтом.

Шкала відношень. В процесі вимірювання ознаці об'єкта ставиться у відповідність деяке дійсне число. Перетворенням в шкалах відношень є довільне лінійно зростаюче відображення. Характерною властивістю цієї шкали є наявність абсолютного нуля. До прикладів ознак, які вимірюються в шкалі відношень, належать відстань у метрах і футах, маса в кілограмах і фунтах, швидкість у кілометрах на годину і вузлах.

Вимірювати ознаки об'єктів кластеризації варто в шкалі відношень або шкалі інтервалів. В цьому випадку результати нечіткої кластеризації мають змістовну інтерпретацію, що відповідає проблемі знаходження нечітких кластерів.

Вектори значень властивостей x_i зручно зображувати у вигляді матриці даних **D** розміру $n \times q$, рядками якої є вектори $\mathbf{x}_i$.

Алгоритм розв'язання задачі нечіткої кластеризації методом нечітких с-середніх. В загальному випадку *задача нечіткого кластерного аналізу за с-методикою* формулюється наступним чином: на основі початкових даних **D** визначити таке нечітке розбиття $\Re(\overline{A}) = \{\overline{A}_k \mid \overline{A}_k \subseteq \overline{A}\}$ множини $\overline{A}$ на задане число c ($c \in N$, $c > 1$) нечітких кластерів $\overline{A}_k$, $k = 1, ..., c$, яке забезпечує екстремум деякої цільової функції $f(\Re(\overline{A}))$ серед всіх нечітких розбиттів.

Для розв'язання цієї задачі треба додатково уточнити вид цільової функції $f(\Re(\overline{A}))$. Для цього розглянемо деякі додаткові поняття.

Вважається, що шукані нечіткі кластери – це нечіткі множини $\overline{A}_k$, які утворюють нечітке покриття множини об'єктів кластеризації $\overline{A}$, для якої виконується наступна умова:

$$\forall a_i \in A: \sum_{k=1}^{c} \mu_{Ak}(a_i) = 1 \qquad (18.1)$$

Виконання (18.1) обумовлене тим, що шукане нечітке покриття має «покривати» звичайну чітку множину об'єктів кластеризації А, що у той же час

є нечіткою множиною $\overline{A}$, для якої значення функцій належності кожного з елементів дорівнює «1».

Разом з кожним нечітким кластером $\overline{A}_k$ розглядаються його типові представники – центри кластера $\mathbf{v}_k$, які обчислюють за формулою:

$$\forall k = 1,...,c,\ \ \forall p_j \in P,\ j = 1,...,q:\ \ v_j^k = \frac{\sum_{i=1}^{n} (\mu_{Ak}(a_i))^m \cdot x_j^i}{\sum_{i=1}^{n} (\mu_{Ak}(a_i))^m} \qquad (18.2)$$

де m – це експонентна вага, яка дорівнює деякому дійсному числу (m>1). Кожний із центрів кластерів $\mathbf{v}_k$=(v_1^k, ..., v_q^k) є вектором у просторі $\Re^q$, якщо всі ознаки виміряні в шкалі відношень.

Як цільову функцію будемо розглядати функцію вигляду:

$$f(A_k, v_j^i) = \sum_{i=1}^{n} \sum_{k=1}^{c} (\mu_{Ak}(a_i))^m \cdot \sum_{j=1}^{q} (x_j^i - v_j^k)^2 \qquad (18.3)$$

де значення m задається залежно від кількості елементів множини A (чим більше елементів містить множина A, тим меншим є значення m).

Задача нечіткої кластеризації за методом нечітких с-середніх формулюється таким чином: для заданих матриці даних **D**, кількості нечітких кластерів c ($c \in N$, $c > 1$) і параметра m визначити матрицю U значень функцій належності об'єктів кластеризації $a_i \in$A (i=1, …, n) нечітким кластерам $\overline{A}_k$ (k = 1, ..., c), які забезпечують мінімум цільової функції (18.3) і задовольняють обмеження (18.1), (18.2), а також додаткові обмеження:

$$\forall k = 1,...,c:\ \sum_{i=1}^{n} \mu_{Ak}(a_i) > 0, \qquad (18.4)$$

$$\forall a_i \in A,\ i = 1,...,n\ \forall k = 1,...,c:\ \mu_{Ak}(a_i) \geq 0. \qquad (18.5)$$

Умова (18.4) унеможливлює появу порожніх нечітких кластерів у шуканій нечіткій кластеризації.

Недоліком задачі нечіткої кластеризації за *с*-методикою є необхідність апріорного задання загальної кількості нечітких кластерів *с*.

В системі MatLab алгоритм розв'язання сформульованої задачі нечіткої кластеризації (18.1) – (18.5) реалізований під назвою FCM (fuzzy *c*-means). Цей алгоритм має ітеративний характер послідовного поліпшення деякого початкового нечіткого розбиття $\Re(\overline{A}) = \{\overline{A}_k \mid \overline{A}_k \subseteq \overline{A}\}$, яке задається користувачем або формується автоматично за деяким евристичним правилом. На кожному кроці перераховуються значення центрів нечітких кластерів та їхніх функцій належності. Алгоритм FCM закінчить свою роботу, коли виконається наперед задане число ітерацій, або коли мінімальна абсолютна різниця між значеннями функцій належності на двох послідовних ітераціях стане меншою за деяке наперед задане число.

Підготовчий етап алгоритму вимагає задати наступні значення параметрів: кількість шуканих нечітких кластерів *c* (*c*∈*N*, *c*>1), максимальну кількість ітерацій алгоритма *s* (*s*∈*N*), параметр збіжності алгоритма ε (ε∈*R*, ε>0), експонентну вагу m (як правило, m=2).

На першій ітерації алгоритму для матриці даних *D* задають деяке нечітке розбиття $\Re(\overline{A}) = \{\overline{A}_k \mid \overline{A}_k \subseteq \overline{A}\}$ на *c* непорожніх нечітких кластерів A_k, яким відповідає сукупність функцій належності $\{\mu_k(a_i), k=1,...,c; a_i \in A, i=1, ..., n\}$.

Отже, алгоритм FCM належить до ітераційних методів групування та включає в себе такі етапи:

Крок 1. Для початкового поточного нечіткого розбиття $\Re(\overline{A}) = \{\overline{A}_k \mid \overline{A}_k \subseteq \overline{A}\}$ розрахувати центри нечітких кластерів v_j^k за формулою (18.2) і значення цільової функції $f(\overline{A}_k, v_j^k)$ за формулою (18.3). Кількість виконаних ітерацій покласти рівною значенню «1».

Крок 2. Сформувати нове нечітке розбиття $\Re'(\overline{A}) = \{\overline{A}_k \mid \overline{A}_k \subseteq \overline{A}\}$ множини об'єктів кластеризації A на *c* непорожніх нечітких кластерів, які характеризуються сукупністю функцій належності $\mu_k'(a_i)$, що визначаються за формулою:

$$\forall k = 1,...,c, \forall a_i \in A,\ i = 1,...,n:\ \mu'_{Ak}(a_i) = \left(\sum_{l=1}^{c} \frac{\left(\sum_{j=1}^{q} (x_j^i - v_j^k)^2 \right)^{\frac{1}{2}}}{\left(\sum_{j=1}^{q} (x_j^i - v_j^l)^2 \right)^{\frac{1}{2}}} \right)^{\frac{2}{m-1}} \Bigg)^{-1} \qquad (18.6)$$

Крок 3. Якщо для деякого $a_i \in$ A існує таке k ($k \in \{1, ..., c\}$), що виконується співвідношення $\sum_{j=1}^{q} (x_j^i - v_j^k)^2 = 0$, то для відповідного нечіткого кластера $\overline{A}_k$ визначити $\mu'_k(a_i)=1$, а для інших $\overline{A}_l$ ($l=1,..., c, l \neq k$) – μ_l *'(*a_i*)=0*.

Якщо ж таких k існує декілька, то для меншого з них визначити $\mu'_k(a_i)=1$, а для інших $l=1,..., c, l \neq k$ – $\mu'_l(a_i)=0$.

Крок 4. Для нового нечіткого розбиття $\Re(\overline{A}) = \{\overline{A}_k \mid \overline{A}_k \subseteq \overline{A}\}$ розрахувати центри нечітких кластерів $\mathbf{v}_j^k$ за формулою (18.2) і значення цільової функції $f'(\overline{A}_k, v_j^k)$ за формулою (18.3).

Крок 5. Якщо кількість виконаних ітерацій перевищує s або ж $\left| f(\overline{A}_k, v_j^k) - f'(\overline{A}_k, v_j^k) \right| \leq \varepsilon$, то в якості шуканої нечіткої кластеризації прийняти нечітке розбиття $\Re'(\overline{A}) = \{\overline{A}_k \mid \overline{A}_k \subseteq \overline{A}\}$ і закінчити виконання алгоритму, інакше вважати поточним нечітким розбиттям $\Re(\overline{A}) = \Re'(\overline{A})$ і перейти до виконання Кроку 2 алгоритму, збільшивши на «1» кількість виконаних ітерацій.

Алгоритм FCM за своїм характером відноситься до наближених алгоритмів пошуку екстремуму для цільової функції (18.3) при наявності обмежень: (18.1), (18.2), (18.4), (18.5). Тому в результаті виконання даного алгоритму визначається локально-оптимальне нечітке розбиття $\Re^*(\overline{A})$, яке описується сукупністю функцій належності $\mu_k(a_i)$, а також центрами кожного з нечітких кластерів $\mathbf{v}_j^k$.

Досвід вирішення прикладних задач нечіткої кластеризації показує, що найбільш ефективний шлях одержання адекватних результатів полягає в

багаторазовому виконанні алгоритму FCM для різних початкових нечітких розбиттів і, якщо не відома кількість нечітких кластерів, для різних значень *c*.

3. Засоби розв'язання задачі нечіткої кластеризації в пакеті Fuzzy Logic Toolbox системи MatLab [3, с. 43–47]. У системі MatLab для розв'язання задачі нечіткої кластеризації на основі алгоритму FCM використовують функцію командного рядка *fcm*, яку викликають 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 функції *fcm* є: 1) *data* – матриця початкових даних **D**, і-й рядок якої зображує інформацію про один об'єкт нечіткої кластеризації $a_i \in A$ в формі вектора $\mathbf{x}_i = [x_1^i \ \ldots \ x_q^i]^T$; 2) *cluster_n* – число шуканих нечітких кластерів *c* ($c \in N$, $c>1$). Вихідними аргументами функції *fcm* є: 1) *center* – матриця центрів шуканих нечітких кластерів $\mathbf{v}_j^k$ (k=1, ..., c; j = 1, ..., q); 2) **U** – матриця значень функцій належності шуканого нечіткого розбиття $\mu_k(a_i)$ (k=1, ..., c, $a_i \in A$); 3) *obj_fcn* – значення цільової функції $f\left(\overline{A}_k, v_j^k\right)$ на кожній ітерації роботи алгоритму.

При виклику функції *fcm* можна використати додаткові аргументи *options*, які мають наступні значення: 1) *options*(1) – експонентна вага *m* для визначення матриці U (за замовчуванням *m*=2); 2) *options*(2) – максимальне число ітерацій s (за замовчуванням s=100); 3) *options*(3) – параметр збіжності алгоритму ε (за замовчуванням ε=0.00001); 4) *options*(4) – інформація про поточну ітерацію, яка відображається на екрані монітора (за замовчуванням це значення дорівнює «1»). Якщо значення додаткового аргумента дорівнює NaN (не число), то для цього аргументу використовується його значення за замовчуванням. Функція *fcm* закінчує свою роботу, коли алгоритм FCM виконає максимальну кількість ітерацій s, або коли різниця між значеннями цільових функцій на двох послідовних ітераціях буде меншою за наперед задане значення параметра збіжності алгоритму ε.

Функція *fcm* реалізована у вигляді m-файлу й, у свою чергу, використовує три інші функції (які також реалізовані у вигляді m-файлів і входять до складу Fuzzy Toolbox): *initfcm* – для формування випадковим чином матриці початкового розбиття; *distfcm* – для розрахунку матриці відстаней між точками даних і центрами кластерів; *stepfcm* – для збереження значень цільової функції й функцій належності об'єктів нечітким кластерам на кожній ітерації роботи алгоритму FCM.

Приклад 18.2. Розглянемо множину даних, які використовують як тестову сукупність об'єктів нечіткої кластеризації. Ці дані зображені у вигляді матриці **D** розміру 140×2, та розташовані в файлі *fcmdata.dat*. Отже, матриця даних **D** відповідає 140 об'єктам, для кожного з яких виконане вимірювання за двома ознаками.

Наведемо послідовність команд, які забезпечують розв'язання задачі нечіткої кластеризації множини даних **D** і візуалізацію отриманих результатів. Використаємо при цьому перший формат запису функції *fcm*, отримаємо:

```
load fcmdata.dat;[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','k');
line(fcmdata(index2,1), fcmdata(index2,2), 'linestyle', ...
                  'none', 'marker', 'O','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)
```

Результат вирішення задачі нечіткої кластеризації для двох нечітких кластерів з використанням цієї послідовності команд зображено на рис. 18.1.

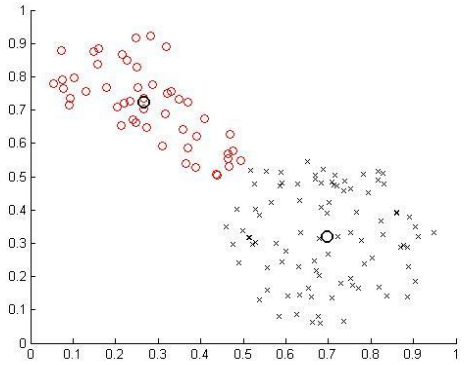


Рис. 18.1. Результат вирішення задачі нечіткої кластеризації для матриці даних з файлу fcmdata.dat

Якщо після запису функції *fcm* не вводити символ «;», то вікно команд буде містити значення координат центрів нечітких кластерів, значення ФН об'єктів нечітким кластерам і значення цільової функції на кожній ітерації роботи алгоритму FCM. Змінимо параметри функції *fcm*, задані за замовчуванням, використавши для цього другий формат її запису з додатковими аргументами, отримаємо:

```
load fcmdata.dat
[center,U,obj_fcn]=fcm(fcmdata, 2, [2.5 1000 0.0000001 1]);
max_U=max(U);
index1=find(U(1,:)== max_U); index2=find(U(2,:)== max_U);
line(fcmdata(index1,1), fcmdata(index1, 2), 'linestyle', ...
                         'none', 'marker', 'o', '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)
```

Результат розв'язання задачі нечіткої кластеризації для двох нечітких кластерів з використанням зазначеної послідовності команд зображено на рис. 18.2. У цьому випадку максимальна кількість ітерацій *s*=1000, експонентна вага *m*=2,5, а параметр збіжності алгоритму ε=0,0000001. Аналіз графіків, зображених на рис. 18.1 і 18.2, показує, що вони майже ідентичні. Це й дозволяє зробити висновок про узгодженість отриманих результатів нечіткої кластеризації.

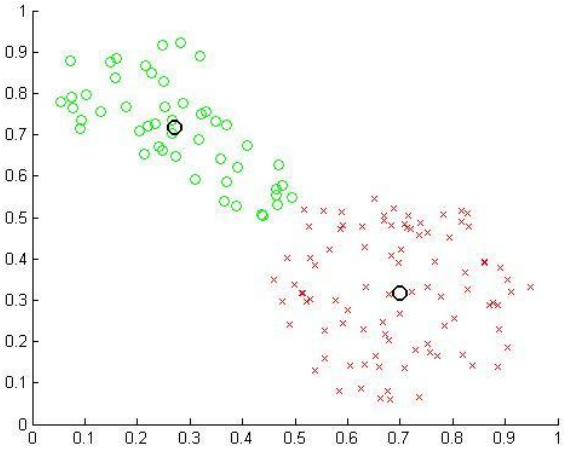


Рис. 18.2. Результат вирішення задачі нечіткої кластеризації із додатковими параметрами функції *fcm*

20. Класифікатори на основі RBF-мережі з використанням FCM кластеризації

Мета роботи: ознайомитися з принципами функціонування та навчання РБФ–мережі, сформованої на основі використанням FCM кластеризації. *Об'єкт дослідження*: РБФ–мережа, сформована на основі використання FCM кластеризації, її використання для розв'язання задач класифікації векторів.

Питання для опрацювання

1. РБФ–мережі, сформованої на основі використання FCM кластеризації [7].

1. Завдання до практичної роботи

1. Розробити класифікатор на основі RBF–мережі з використанням FCM кластеризації для діагностики.

Постановка задачі

На основі навчальної вибірки сформовано кластери і генерує функцію c: $\Re^n \to L$, де L – множина міток класів. Для бінарного класифікатора маємо L={–1 (або 0), 1}. Якщо на вхід подається вектор $\mathbf{x}=\mathbf{x}^i=[x_1 \ldots x_n]^T$, $i=1, \ldots, Q$, то його вихід $c(\mathbf{x})$ дорівнюватиме значенню «1», якщо мітка класу більше значення «0», та «–1», якщо мітка класу менше або дорівнює значенню «0».

Нехай задано навчальну вибірку у вигляді пар даних вхід-ціль: $\{\mathbf{x}^1, t^1\},\ldots, \{\mathbf{x}^Q, t^Q\}$, яка генерується функцією $t^i=f(\mathbf{x}^i)$, $i=1, \ldots, Q$, де $\mathbf{x}^i=[x_1^i \ \ldots \ x_n^i]^T$ – вектор вхідних даних з елементами $x_j^i \in \Re$; t^i – бажаний відгук. Функція $f(\cdot)$ вважається невідомою, але задано множину її реалізацій: $T=\left\{(x_1^i, ..., x_n^i, t^i),\ i=1, 2, ..., Q, n>1\right\}$.

Побудувати РБФ–мережу для визначення функції $F(\mathbf{w}, \mathbf{x}^i)$, яка апроксимує функцію $f(\mathbf{x})$, описуючи перетворення вхідного сигналу у вихідний, і задовольняє умову $\frac{1}{Q}\sum_{i=1}^{Q}\left|F(\mathbf{w},\mathbf{x}^i)-t^i\right|<\varepsilon,$ де ε – деяке додатне число, яке називається нев'язкою. Ініціалізацію РБФ-мережі (функції $F(\mathbf{w}, \mathbf{x}^i)$) виконано на основі Дерева розв'язків. Вагові коефіцієнти вихідного шару визначаємо на основі псевдооберненого правила.

Порядок виконання роботи

Крок 1. Сформувати кластери на основі використання методу FCM кластеризації.

Крок 2. Пов'язати кожний кластер із одним РБФ-нейроном, його параметри (центр, дисперсія) залежать від розташування і вимірів кластера.

Крок 3. Організувати нейрони в один прихований шар. Визначити вагу вихідного шару.

Крок 4. Перевірити РБФ-мережу на здатність розв'язувати задачу класифікації на множині навчання та тестування.

2. Контрольні запитання та завдання

1. Опишіть алгоритм формування РБФ-мережі, сформованої на основі методу FCM кластеризації.

Варіанти індивідуальних завдань: комп'ютерний практикум № 7.

Аудиторна робота

Приклад 20.1: MatLab. Реалізація алгоритму формування класифікатора у вигляді РБФ-мережі, сформованої на основі методу FCM кластеризації.

1. Без нормуванням даних

```
clc; clear all; close all;
f = fopen('data.txt');
input_data = fscanf(f,'%d;%f;%f;%f;%f;%f;%f;%f;%f;%f;%d',[11
inf])'; fclose(f);
train_data = input_data(1:10,2:10);
train_t = input_data(1:10,11);
test_data = input_data(11:16,2:10); S1=4;
test_t = input_data(11:16,11)';
d = dispersion(train_data); d1=dispersion(test_t);

%кластеризація [centers, U] = fcm(train_data,S1);
centers = f_c_means(train_data,S1);
maxU=max(U);
for j = 1:10
    for i = 2
     x1=find(U(i,:)==maxU); hold on;
plot(train_data(index1, j), train_data(index1, j+1), 'or');
plot(centers(I,j), centers(I,j+1),'xb', 'MarkerSize', 15,
'LineWidth', 3); hold off;
end; end;
[centers_n, centers_m]=size(centers);
train_n = size(train_data);
% обчислення матриці інтерполяції G
G = zeros(train_n(1),centers_n);
for i = 1:train_n(1)
    for j = 1:centers_n
     G(i,j) = gauss(train_data(i,:),centers(j,:),centers_m,d);
end; end
% Обчислення вектора ваги
Gplus = inv(G'*G)*G'; W = Gplus*train_t; size_w = size(W);

% класифікація на даних навчання
test_n = size(test_data); test_y = zeros(1,test_n(1));
for i = 1:train_n(1)
  y = 0;
  for k = 1:size_w
    y=y+W(k)*gauss(train_data(i,:),centers(k,:),centers_m,d);
  end
```

```
  if y <= 0
      test_y(i)=-1;
  else
      test_y(i)=1;
end; end
error1=0;
for i=1:test_n(1)
    if not(train_t(i)==test_y(i))
        error1 = error1 + 1;
end; end
% класифікація на тестових даних
test_n = size(test_data); test_y = zeros(1,test_n(1));
for i=1:test_n(1)
  y=0;
  for k = 1:size_w
    y=y+W(k)*gauss(test_data(i,:),centers(k,:),centers_m,d);
  end
  if y <= 0
      test_y(i)=-1;
  else
      test_y(i)=1;
end; end
% порівняння тестових даних з справжніми
error2= 0;
for i = 1:test_n(1)
    if not(test_t(i)== test_y(i))
        error2=error2 + 1;
end; end;
error2,error1

function centers = f_c_means(data,cluster_n)
    e = 0.1; [n,m]=size(data);
    centers = zeros(cluster_n,m);
    % ініціалізація матриці належності
    O = zeros(n,cluster_n);
    for i = 1:n
        k = 1;
        for j = 1:cluster_n
            r = k*rand; k = k - r;
            O(i,j)= r;
    end; end
% кластеризація
    cond = false;iteration = 1;
    d = zeros(cluster_n,n);
    while(not(cond) && iteration <= 100)
        disp(['Iteration ' num2str(iteration)]);
        O_old = O;
    % оновлення матриці центрів
        for i=1:cluster_n  % кожен кластер
            for k=1:m  % кожен атрибут
                c=0; z=0;
                for j=1:n
```

```
                c=c + (O(j,i)^2)*data(j,k); % чисельник
                z=z + O(j,i)^2;  % знаменник
              end;
              centers(i,k) = c/z;
       end; end;
    % оновлення матриці відстаней d
       for i=1:cluster_n
           for j=1:n
               c=0;
               for l=1:m
                 c=c+(data(j,l)-centers(i,l))^2;
               end
               d(i,j) = sqrt(c);
       end; end
    % оновлення матриці O
       for i=1:cluster_n
           for j=1:n
             c=0;
             if d(i,j)==0
              O(j,i)=1;
             else
              for k=1:cluster_n
               c = c + (d(i,j)/d(k,j))^2;
              end;
              O(j,i)=1/c;
             end;
       end; end;
       % перевірка умови виходу
       if sum(abs(O-O_old)) <= e
          cond=true;
       end
       iteration=iteration + 1;
end; end
function dispersion = dispersion(data)
[n,m] = size(data); dispersion = zeros(1,m);
for k=1:m
    p=0;
    for i=1:n
        p=p+data(i,k);
    end
    p = p/n; z = 0;
    for i=1:n
        z=z+(data(i,k)-p)^2;
    end; dispersion(k) = z/n;
end; end
function val = gauss(x,center,m,dispersion)
    val = 1;
    for i = 1:m
      val = val + ((x(i)-center(i))^2)/dispersion(i);
    end;
    val = exp(-sqrt(val));
end
```

Фрагмент Бази даних пацієнтів щодо захворювання на рак молочної залози

Дані навчання	Дані тестування
1000025;5;1;1;1;2;1;3;1;1;-1	1035283;1;1;1;1;1;1;3;1;1;-1
1002945;5;4;4;5;7;10;3;2;1;-1	1036172;2;1;1;1;2;1;2;1;1;-1
1015425;3;1;1;1;2;2;3;1;1;-1	1041801;5;3;3;3;2;3;4;4;1;1
1016277;6;8;8;1;3;4;3;7;1;-1	1043999;1;1;1;1;2;3;3;1;1;-1
1017023;4;1;1;3;2;1;3;1;1;-1	1044572;8;7;5;10;7;9;5;5;4;1
1017122;8;10;10;8;7;10;9;7;1;1	1047630;7;4;6;4;6;1;4;3;1;1
1018099;1;1;1;1;2;10;3;1;1;-1	
1018561;2;1;2;1;2;1;3;1;1;-1	
1033078;2;1;1;1;2;1;1;1;5;-1	
1033078;4;2;1;1;2;1;2;1;1;-1	

Результат

```
error2 =  2; error1 =  0
centers =
3.6598 1.3115 1.2477 1.3699 2.0150 1.4428 2.7831 1.0883 1.0171
4.6009 3.9548 3.9551 4.2030 6.0815 9.5103 2.9993 2.2746 1.0013
7.8550 9.8456 9.8457 7.5666 6.7483 9.6527 8.6278 6.9737 1.0002
2.2180 1.3611 1.4082 1.0422 2.0508 1.9615 1.4882 1.2993 4.0005

d = 4.0000 10.0000 10.0000 5.2100 3.8900 15.6900 4.0100 5.6100
1.4400
W =-2.4666
   -3.4758
    2.8737
   -3.8433
```

2. Із нормуванням

```
clc; clear all; close all;
f=fopen('data.txt');
input_data=fscanf(f,'%d;%f;%f;%f;%f;%f;%f;%f;%f;%f;%d',[11 inf])';
fclose(f); train_data=input_data(1:10,2:10);
train_t=input_data(1:10,11);
test_data=input_data(11:16,2:10);
train_data=norm16(train_data);
test_data=norm16(test_data); % S1=4;
test_t=input_data(11:16,11)';
d=dispersion(train_data); d=0.25*d;
% кластеризація [centers, U] = fcm(train_data,4);
centers = f_c_means(train_data,4);
[centers_n, centers_m] = size(centers);
train_n = size(train_data);
% обчислення матриці інтерполяції G
G=zeros(train_n(1),centers_n);
for i=1:train_n(1)
    for j=1:centers_n
     G(i,j)=gauss(train_data(i,:),centers(j,:),centers_m,d);
end; end
% обчислення вектора ваги
```

```
Gplus = inv(G'*G)*G'; W = Gplus * train_t; size_w = size(W);
% класифікація на даних навчання
test_n = size(test_data); test_y = zeros(1,test_n(1));
for i = 1:train_n(1)
  y = 0;
  for k = 1:size_w
   y=y+W(k)*gauss(train_data(i,:),centers(k,:),centers_m,d);
  end
  if y <= 0
      test_y(i)=-1;
  else
      test_y(i)=1;
end; end
error1 = 0;
for i = 1:test_n(1)
    if not(train_t(i) == test_y(i))
        error1 = error1 + 1;
end; end
% класифікація на тестових даних
test_n = size(test_data); test_y = zeros(1,test_n(1));
for i = 1:test_n(1)
  y = 0;
  for k = 1:size_w
     y=y+W(k)*gauss(test_data(i,:),centers(k,:),centers_m,d);
  end
  if y <= 0
      test_y(i)=-1;
  else
      test_y(i)=1;
end; end
%порівняння тестових даних з справжніми
error2= 0;
for i = 1:test_n(1)
    if not(test_t(i) == test_y(i))
        error2 = error2 + 1;
end; end
error2,error1
function xx = norm16(x)
% x=[6 4 3 2 3 4 5 6 7 8 9 0; 5 7 3 5 2 5 7 8 8 6 4 4]; x=x';
 r=size(x); dl=zeros(r(1),1);xx=zeros(r(1),r(2));
 for i=1:r(1)
   dl(i)=norm(x(i,:));
 end;
 for i=1:r(1)
    for j=1:r(2)
     xx(i,j)=x(i,j)/dl(i,1);
 end;end; end
```

Результат

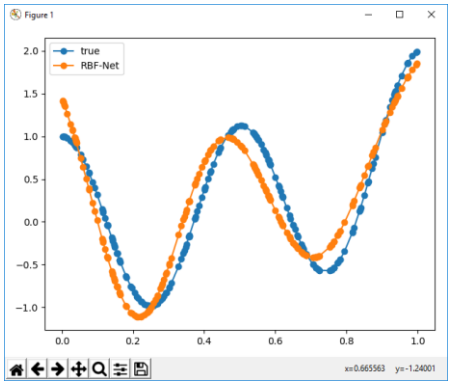
```
error2 = 1; error1 =  1
```

Приклад 17.2: python. Реалізація алгоритму формування апроксиматора у вигляді РБФ-мережі, сформованої на основі методу FCM кластеризації.

```
import numpy as np
import matplotlib.pyplot as plt

def rbf(x, c, s):
    return np.exp(-1/(2*s**2)*(x-c)**2)

def fcm(X, k):
    clusters = np.random.choice(np.squeeze(X), size=k)
    prevClusters = clusters.copy()
    stds = np.zeros(k)
    converged = False
    while not converged:
        distances=np.squeeze(np.abs(X[:,np.newaxis]-
clusters[np.newaxis, :]))
        closestCluster = np.argmin(distances, axis=1)
        for i in range(k):
            pointsForCluster = X[closestCluster == i]
            if len(pointsForCluster) > 0:
                clusters[i]=np.mean(pointsForCluster, axis=0)
        converged=np.linalg.norm(clusters-prevClusters)<1e-6
        prevClusters = clusters.copy()
    distances=np.squeeze(np.abs(X[:,np.newaxis]-
clusters[np.newaxis, :]))
    closestCluster = np.argmin(distances, axis=1)
    clustersWithNoPoints = []
    for i in range(k):
        pointsForCluster = X[closestCluster == i]
        if len(pointsForCluster) < 2:
            clustersWithNoPoints.append(i)
            continue
        else:
            stds[i]=np.std(X[closestCluster==i])
    if len(clustersWithNoPoints)>0:
        pointsToAverage=[]
        for i in range(k):
            if i not in clustersWithNoPoints:
                pointsToAverage.append(X[closestCluster==i])
        pointsToAverage = np.concatenate(pointsToAverage).ravel()
        stds[clustersWithNoPoints]=
np.mean(np.std(pointsToAverage))
    return clusters, stds

class RBFNet(object):
  def __init__(self,k=2,lr=0.01,epochs=100,rbf=rbf,
inferStds=True):
        self.k=k; self.lr = lr
        self.epochs=epochs; self.rbf = rbf
        self.inferStds = inferStds
```

```
        self.w = np.random.randn(k)
        self.b = np.random.randn(1)
  def fit(self, X, y):
        if self.inferStds:
            self.centers, self.stds=fcm(X, self.k)
        else:
            self.centers, _ = fcm(X, self.k)
            dMax = max([np.abs(c1-c2) for c1 in self.centers for
c2 in self.centers])
            self.stds=np.repeat(dMax / np.sqrt(2*self.k), self.k)
        # training
        for epoch in range(self.epochs):
          for i in range(X.shape[0]):
            a=np.array([self.rbf(X[i],c,s)        for        c,s,in
zip(self.centers, self.stds)])
            F = a.T.dot(self.w) + self.b
            loss = (y[i] - F).flatten() ** 2
            print('Loss: {0:.2f}'.format(loss[0]))
            error = -(y[i] - F).flatten()
            self.w = self.w - self.lr * a * error
            self.b = self.b - self.lr * error
  def predict(self, X):
        y_pred = []
        for i in range(X.shape[0]):
            a=np.array([self.rbf(X[i],c,s)    for    c,    s,    in
zip(self.centers, self.stds)])
            F=a.T.dot(self.w)+self.b; y_pred.append(F)
        return np.array(y_pred)

# sample inputs and add noise
NUM_SAMPLES = 200
X = np.random.uniform(0., 1., NUM_SAMPLES)
X = np.sort(X, axis=0); y = np.cos(np.pi * 4 * X) + X**3
rbfnet = RBFNet(lr=1e-2, k=2, inferStds=True)
rbfnet.fit(X, y); y_pred = rbfnet.predict(X)
plt.plot(X, y, '-o', label='true')
plt.plot(X, y_pred, '-o', label='RBF-Net')
plt.legend(); plt.tight_layout(); plt.show()
```

Результат роботи програми: y=cos(pi * 4 * x) + x**3



Теоретичний матеріал

РБФ-мережі, сформованої на основі використання FCM кластеризації [7]. RBF–мережі, зазвичай, навчаються на основі пар вхід-вихід $\{\mathbf{x}(t), y(t), t=1, ..., T\}$. Далі обирається центр РБФ-нейрона у прихованому шарі. Цей етап виконується різними способами: центри можна 1) випадково відібрати з деякого набору прикладів, 2) визначити за допомогою кластеризації методом k–середніх або FCM; 3) визначити за допомогою Дерева розвязків (комп'ют. практикум № 14). Розглянемо метод визначення центрів за допомогою кластеризації FCM (комп'ютерний практикум № 3) .

Метод нечіткої кластеризації C-means дозволяє розбити множину векторів (точок) із простору R^k на задану кількість нечітких підмножин. Особливістю методу є використання нечіткої матриці належності **U** з елементами μ_{ij} (i=1, …, N - кількість об'єктів класифікації, j=1, …, c – кількість кластерів), які визначають належність i-го елемента вихідної множини векторів j–ому кластеру. Кластери описуються своїми центрами $\mathbf{v}_j \in R^k$, j = 1, …, c. Алгоритм виконує такі кроки:

Крок 1. *Ініціалізація*. Задаються t=0; кількість кластерів c; порогове значення похибки ε, що завершує ітераційний процес. Створюється початкова матриця належності **U**, значення якої заповнюються випадковим чином, але з умовою що сума її елементів для кожного об'єкту по всім кластерам дорівнює одиниці.

Крок 2. *Корегування центрів кластерів*. Розрахунок центрів кластерів відбувається на основі матриці належності **U** за формулою: $v_{jl} = \dfrac{\sum_{j=1}^{N} \mu_{ij}^2 x_{il}}{\sum_{j=1}^{N} \mu_{ij}^2}$,

j=1, …, c; l= 1, ..., k.

Крок 3. *Корегування матриці ступенів належності.* Обчислення відстані між елементами x_j та окремими кластерами i: i=1, …, N; j=1, …, c

$$d_{ij}(t) = \left\| \mathbf{x}_i - \mathbf{v}_j \right\| = \sqrt{\sum_{l=1}^{k} \left(x_{il} - v_{jl} \right)^2}.$$

Відновлення матрицю належності об'єктів до кластерів за формулою:

якщо $d_{ij} \neq 0$, то $\mu_{ij}(t+1) = \dfrac{1}{\sum_{k=1}^{c} \left[\dfrac{d_{ij}(t)}{d_{ik}(t)} \right]^2}$, інакше $\mu_{ij} = 1$.

Перевірка умови закінчення процесу кластеризації за формулою:

$$\sum_{i=1}^{N} \sum_{j=1}^{c} \left| \mu_{ij}(t+1) - \mu_{ij}(t) \right| \leq \varepsilon.$$

Якщо умова виконується, то перейти до кроку 4, інакше на крок 2.

Крок 4. Після того як центри зафіксовані, вагу, яка мінімізує похибку на виході мережі, обчислюють за допомогою псевдооберненого правила $\mathbf{w}=\mathbf{G}^{+}\cdot\mathbf{t}=(\mathbf{G}^{T}\cdot\mathbf{G})^{-1}\cdot\mathbf{G}^{T}\cdot\mathbf{t}$.

21. Розв'язання задачі екстраполяції даних з використанням РБФ- та нейронечіткої мереж

Мета роботи: ознайомитися з можливістю розв'язання задачі екстраполяції даних на основі РБФ– та Anfis–мереж. *Об'єкт дослідження:* процес екстраполяції даних на основі РБФ– та Anfis–мережі.

Питання для опрацювання

1. Екстраполяція нечіткої моделі із одним входом [7].
2. Розв'язання задачі екстраполяції даних

1. Задання до практичної роботи

1. Вивчити теоретичний матеріал.
2. Послідовно виконати такі завдання до практичної роботи «Розробка та використання моделей для розв'язання задачі екстраполяції даних (у вигляді m-файлів)»:

2.1. Спроектувати РБФ–мережу та Anfis–мережу для розв'язання задачі екстраполяції даних.

2.2. Визначити похибку екстраполяції.

Постановка задачі екстраполяції даних. Розглянемо відображення $d = f(x)$, де $\mathbf{x} = [x_1 \ ... \ x_N]^{\mathrm{T}}$ – вектор вхідних даних, $x_j \in X_j \subset Y_j$, $X_j \neq Y_j$ (j=1, …, N), $\mathbf{d} = [d_1 \ ... \ d_M]^{\mathrm{T}}$ – вектор вихідних даних. Вектор–функція $f(\cdot)$ вважається невідомою, але задано множину реалізацій функції

f: $T = \{(x_1^i,...,x_N^i, d_1^i,...,d_M^i) : 1 \le i \le p;\ N \ge 1;\ M \ge 1\}$.

Необхідно побудувати мережу, яка екстраполює функцію $f(\cdot)$, та задовольняє умову: $\frac{1}{p}\sum_{i=1}^{p}\left|F(\mathbf{x}^i) - \mathbf{d}^i\right| < \varepsilon$, де $F(\cdot)$ – функція, яка описує перетворення вхідного сигналу у вихідний, $\mathbf{x}^i \in Y \backslash X$, $X = X_1 \times \ldots \times X_N$, $Y = Y_1 \times \ldots \times Y_N$, ε – деяке додатне число, яке називається нев'язкою.

В рамках даної роботи використати РБФ–мережу та нечітку нейронну мережу.

Задачі для самостійного розв'язання: комп'ютерний практикум № 4

Аудиторна робота

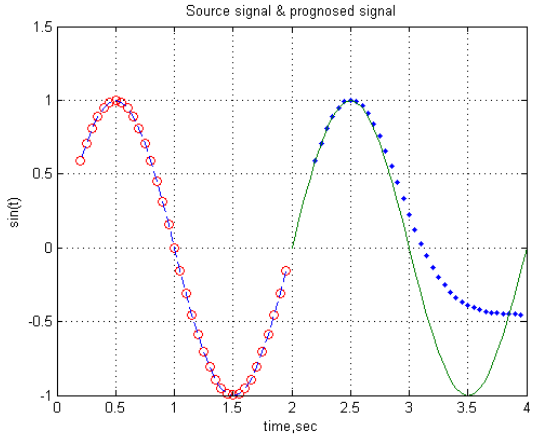
Приклад 21.1: використання РБФ-мережі:

```
close all; clear all;
t=0:0.05:2; y=sin(pi*t);
% Формування вхідного та цільового векторів
k=length(t);
for i=1:k-5
 P4(1,i)=t(1,i); P3(1,i)=t(1,i+1);
 P2(1,i)=t(1,i+2); P1(1,i)=t(1,i+3); T(1,i)=y(1,i+4)
end;
```

```
P=[P4;P3;P2;P1]; net=newrbe(P,T);
hidden_neuron_quantity_for_newrbe=net.layers{1}.size
A=sim(net,P); % моделювання
t_test=2.0:0.05:4;y_test=sin(pi*t_test);k1=length(t_test)
for i=1:k1-5
 P4(1,i)=t_test(1,i); P3(1,i)=t_test(1,i+1)
 P2(1,i)=t_test(1,i+2); P1(1,i)=t_test(1,i+3)
end
t_T1=5:k1-1; P_test=[P4;P3;P2;P1],
A_test=sim(net,P_test) %моделювання
figure (1)
plot (t(5:k-1),A,'or',t(5:k-1),T,'--',t_test(t_T1), ...
 A_test, '.b', t_test,y_test);
grid on; xlabel('time,sec');ylabel('sin(t)');
title('Source signal & prognosed signal');
e=sum(A_test-y_test(5:k1-1))/length(A_test-y_test(5:k1-1))
```

Результат роботи мережі:

e = 0.1926 `[P1;P2;P2;P4]` – нев'язка екстраполяції даних



net = Neural Network

name: **'Radial Basis Network, Exact'**
efficiency: .cacheDelayedInputs, .flattenTime,
.memoryReduction
userdata: (your custom info)
dimensions:
 numInputs: 1
 numLayers: 2
numOutputs: 1
numInputDelays: 0
numLayerDelays: 0
 numFeedbackDelays: 0
 numWeightElements: 217
sampleTime: 1
connections:
 biasConnect: [1; 1]
 inputConnect: [1; 0]
 layerConnect: [0 0; 1 0]
 outputConnect: [0 1]

```
subobjects:
inputs: {1x1 cell array of 1 input}
layers: {2x1 cell array of 2 layers}
outputs: {1x2 cell array of 1 output}
biases: {2x1 cell array of 2 biases}
inputWeights: {2x1 cell array of 1 weight}
layerWeights: {2x2 cell array of 1 weight}
functions:
  adaptFcn: (none)
adaptParam: (none)
  derivFcn: 'defaultderiv'
 divideFcn: (none)
   divideParam: (none)
divideMode: 'sample'
   initFcn: 'initlay'
performFcn: (none)
  performParam: (none)
  plotFcns: {}
plotParams: {1x0 cell array of 0 params}
  trainFcn: (none)
trainParam: (none)
weight and bias values:
IW: {2x1 cell} containing 1 input weight matrix
LW: {2x2 cell} containing 1 layer weight matrix
b: {2x1 cell} containing 2 bias vectors
methods:
adapt: Learn while in continuous use
configure: Configure inputs & outputs
gensim: Generate Simulink model
init: Initialize weights & biases
perform: Calculate performance
sim: Evaluate network outputs given inputs
train: Train network with examples
view: View diagram
unconfigure: Unconfigure inputs & outputs

evaluate:   outputs = net(inputs)
```

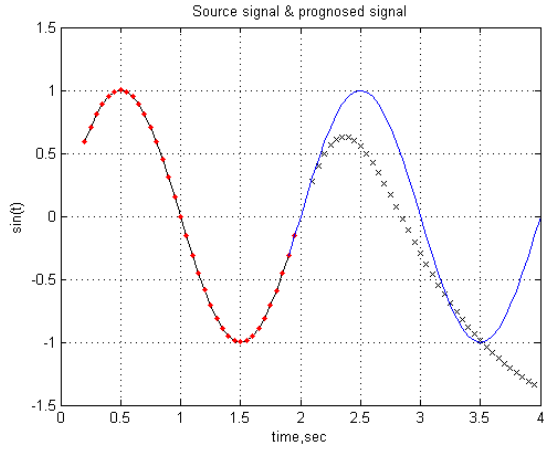
Приклад 21.2. Використання нейронечіткої Anfis-мережі

```
close all; clear all;
t=0:0.05:2; y=sin(pi*t);k=length(t)
%Формування вхідного вектору
for i=1:k-5
  T(i)=y(1,i+4);
  P4(i)=t(1,i);   P3(i)=t(1,i+1);
  P2(i)=t(1,i+2); P1(i)=t(1,i+3);
end
trnData=[P1;P2;P3;P4; T]'; %Формування векторів входу, цілей
numMFS=3;
% кількість ф-цій належності, пов’язаних з кожним виходом
 mfType='gbellmf'; %тип функції належності
```

```
 epoch_n=40;
% створення FIS структури
 in_fismat=genfis1(trnData,numMFS,mfType);
 out_fismat=anfis(trnData,in_fismat,epoch_n); % навчання
 R1=evalfis([P1;P2;P3;P4],out_fismat)
 t_test=1.9:0.05:4;
 y_test=sin(pi*t_test);

 k1=length(t_test)
 for i=1:k1-5
  P4(1,i)=t_test(1,i); P3(1,i)=t_test(1,i+1)
  P2(1,i)=t_test(1,i+2); P1(1,i)=t_test(1,i+3)
 end
 t_T1=5:k1-1; P_test=[P1;P2;P3;P4]
 R2=evalfis(P_test,out_fismat)  % моделювання
 figure (2)
 plot (t(5:k-1),T,'-k',t(5:k-1),R1,'.r',t_test(t_T1), ...
R2, 'xk', t_test,y_test);
 grid on; xlabel('time,sec');ylabel('sin(t)');
 title('Source signal & prognosed signal');
 e=sum(R2-y_test(5:k1-1)')/length(R2-y_test(5:k1-1)')
```

Результат моделювання:

e= -0.3118 - нев'язка екстраполяції даних



Теоретичний матеріал

Екстраполяція нечіткої моделі із одним входом [7]. Екстраполяція дозволяє передбачити поведінку систем за межами поточних областей їх функціонування. Екстраполяція моделі має вигляд припущення, яке не може бути обгрунтовано внаслідок відсутності в момент прийняття рішень інформації про поведінку системи у новій області.

Розглянемо систему з одним входом x, модель якої має вигляд $y=f(x)$. Якщо модель є неперервною і має неперервні похідні в граничних точках області визначення X=[a, b], то використовуючи розкладання в ряд Тейлора, наближене значення $f^*(x)$ в точці $x=b+h$, розташованій в близькості від області, де достовірність моделі підтверджена результатами вимірювань (позначимо цю область ОД), можна обчислити за формулою:

$$f(b+h) \cong f(b) + \frac{h}{1!} f'(b) + \frac{h^2}{2!} f»(b) + \ldots + \frac{h^n}{n!} f^{(n)}(b),$$

де n – порядок екстраполяції. Найпростішим варіантом екстраполяції є екстраполяція нульового порядку вигляду: $f^*(b+h) = f(b)$, $f^*(a-h) = f(a)$. Приклад такої екстраполяції наведено на рис. 21.1. Єдиною інформацією про область достовірності моделі є граничне значення функції $f(a)$ або $f(b)$.

Екстраполяцію першого порядку виражають формулами: $f^*(b+h)=f(b)+h\,\dot{f}(b)$, $f^*(a-h) = f(a) - h\,\dot{f}(a)$, її приклад наведено на рис. 21.2.

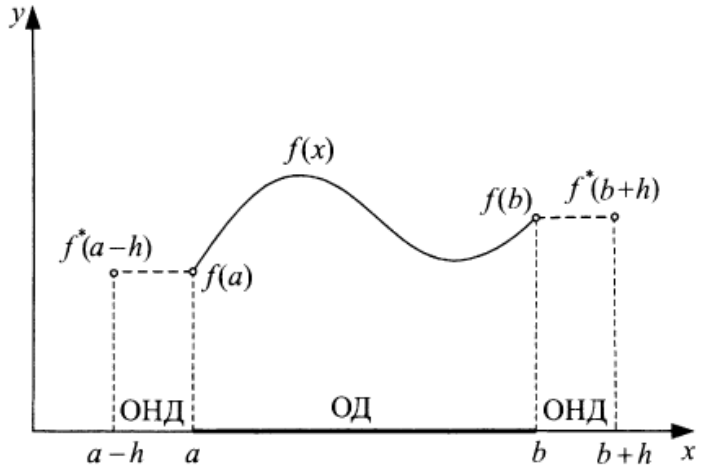


Рис. 21.1. Екстраполяція нульового порядку функції $f(x)$: [a, b] – область достовірності функції; x <a, x> b – області, в яких достовірність функції не підтверджена (ОНД)

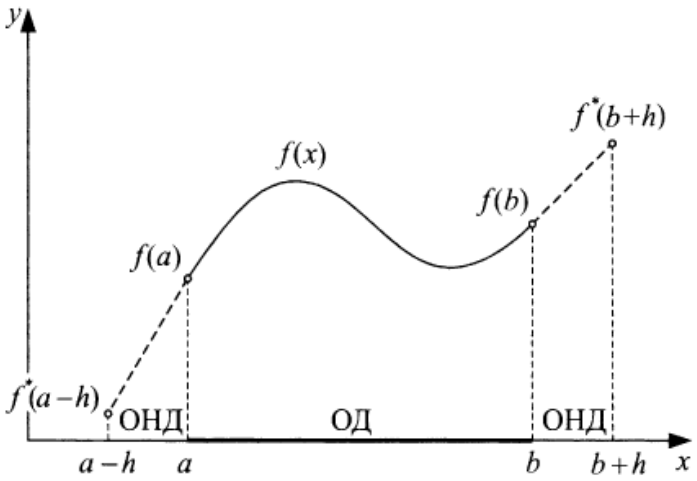


Рис. 21.2. Екстраполяція першого порядку: ОД – область, де достовірність моделі підтверджена вимірами, ОНД – область, в якій достовірність моделі не підтверджена

Екстраполяція першого порядку використовує інформацію не тільки про граничне значення функції $f(a)$ і $f(b)$, але також про значення похідної $\dot{f}(a)$ або $\dot{f}(b)$ на межі області достовірності.

Екстраполяція другого порядку виражається формулами:

$$f^*(b+h) = f(b) + h\dot{f}(b) + \frac{h^2}{2}\ddot{f}(b),$$
$$f^*(a-h) = f(a) - h\dot{f}(a) + \frac{h^2}{2}\ddot{f}(a).$$

Розглянемо спрощену задачу про приріст прибутку.

Приклад 21.3. Концерн супермаркетів протягом декількох років інвестував різні суми в розвиток своєї мережі, що в результаті давало йому щороку різні значення приросту прибутку (табл. 21.1).

Таблиця 21.1

Капіталовкладення концерну та їх фінансові результати

Рік	1997	1998	1999	2000
Капіталовкладення CE, млн. дол.	100	150	210	230
Приріст прибутку ΔE, млн. дол.	220	270	300	?

Керівництво концерну вважає за можливе в 2000 р. інвестувати в будівництво нових супермаркетів суму в 230 млн. дол. Який приріст прибутку ΔЕ можна при цьому очікувати?

На основі даних табл. 21.1 можна побудувати просту нечітку модель, яка містить такі три правила:

Якщо (капіталовкладення низькі), То (приріст прибутку низький)
Якщо (капіталовкладення середні), То (приріст прибутку середній)
Якщо (капіталовкладення високі), То (приріст прибутку високий)

Для окремих областей ОД можна отримати залежність ΔЕ=*f*(СЕ), яка відповідає нечіткій моделі (рис. 21.3): ΔЕ=СЕ+120 для 100 ≤ СЕ ≤ 150, ΔЕ=0,5·СЕ+195 для 150 ≤ СЕ ≤ 210.

При використанні в межах ОНД функцій належності «низький», «середній» і «високий» (рис. 21.3), буде виникати ефект насичення, і поверхня моделі матиме вигляд: ΔЕ=220 для СЕ<100, ΔЕ=300 для СЕ>210.

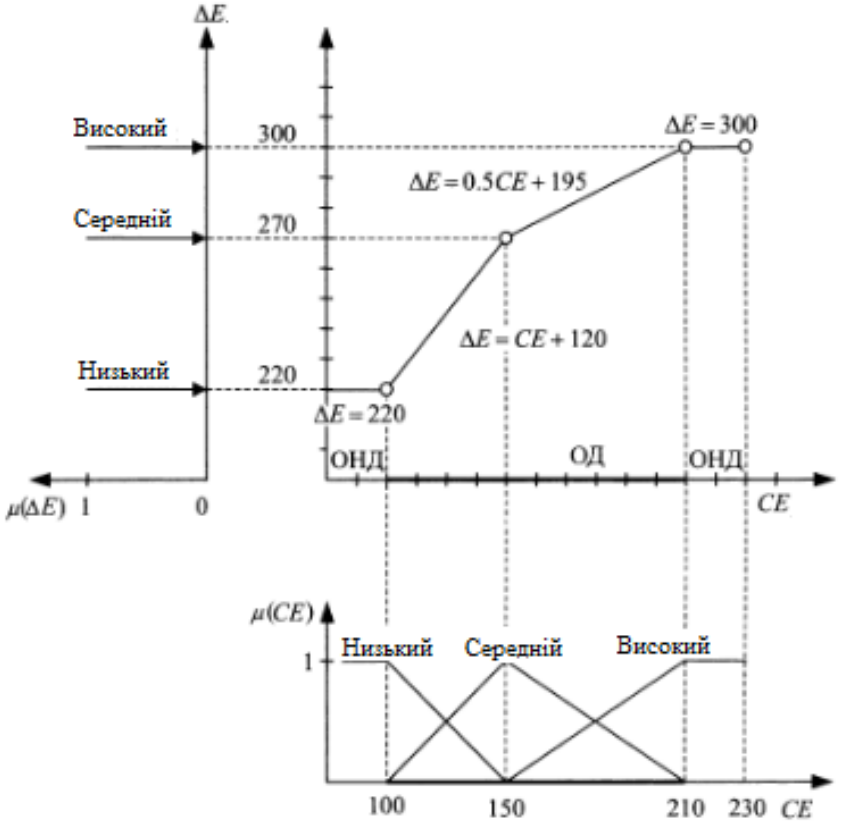


Рис. 21.3. Відображення «вхід-вихід» нечіткої моделі, яка визначає взаємозв'язок між величиною капіталовкладень СЕ і приростом прибутку ΔЕ із функціями належності, значення яких обмежені інтервалом [0, 1]

Ця ситуація відповідає екстраполяції нульового порядку, яка використовує в ОНД–областях тільки інформацію про значеннях функції на межі області достовірності. В результаті застосування цього типу екстраполяції при обсязі капіталовкладень CE=230 млн. дол. прогнозоване значення приросту прибутку буде дорівнювати значенню, як і у випадку CE=210 млн. дол. (рис. 21.3).

Введемо в нечітку модель новий тип ФН вхідних значень (рис. 21.4). В результаті використання цих функцій в межах області достовірності вигляд відображення «вхід–вихід» залишається таким же, як і в разі використання звичайних ФН, в той час як в ОНД-областях формула екстраполяції набуває такого вигляду: $\Delta E=CE+120$ для $CE < 100$, $\Delta E=0{,}5\cdot CE+ 195$ для $CE > 210$.

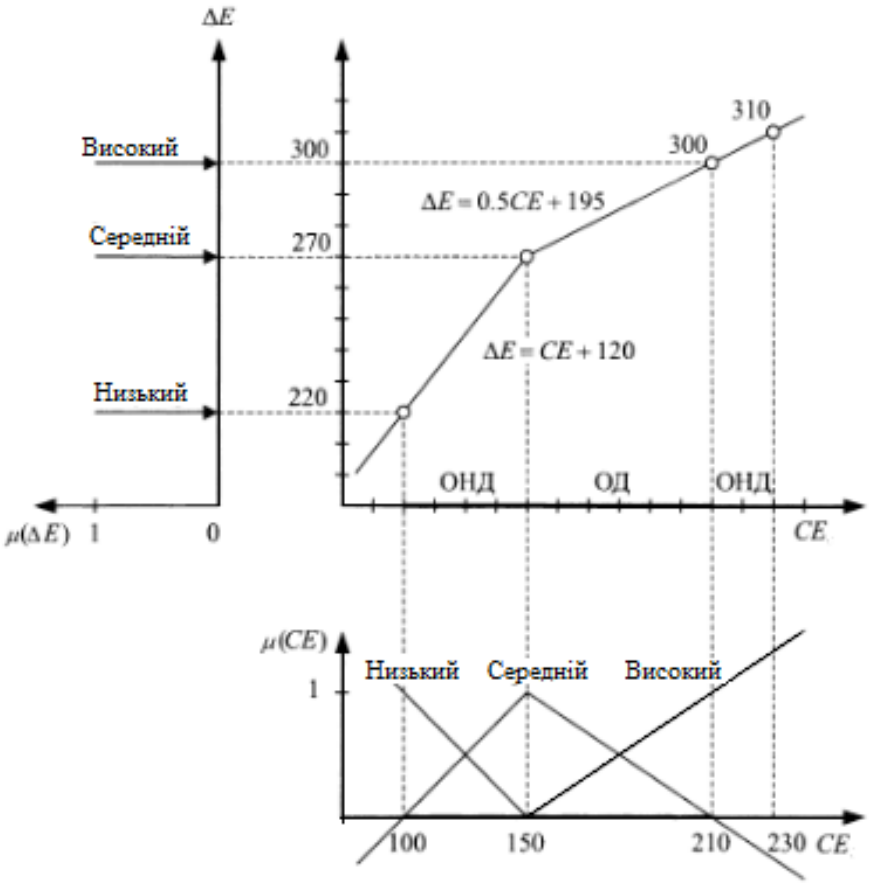


Рис. 21.4. Відображення «вхід–вихід» нечіткої моделі, яка визначає взаємозв'язок між величиною капіталовкладень CE і приростом прибутку ΔE, в цій моделі значення крайніх ФН не обмежені інтервалом [0, 1]

Відповідно до отриманої моделі прогнозу при обсязі капіталовкладень CE = 230 млн. дол. приріст прибутку ΔE=310 млн. дол., що перевершує отримане в рамках попередньої моделі значення 210 млн. дол. (рис. 21.4).

Як показано на рис. 3, нечітка модель з традиційним видом ФН нечутлива до змін входів в ОНД-областях, де значення ФН є постійними (рівними «0» або «1»). Модель, зображена на рис. 4, є чутливою до змін входів як в межах області достовірності, так і в ОНД-областях. У ОНД-області модель використовує інформацію не тільки про граничне значення вихідного параметру ΔE = 300 млн. дол., але також про величину нахилу поверхні в прилеглій до межі частині області достовірності, що відповідає екстраполяції першого порядку.

Тема 3. Моделі із використанням генетичного алгоритму

22. Бінарний генетичний алгоритм

Мета роботи: ознайомитися з принципами функціонування бінарного генетичного алгоритму (ГА). *Об'єкт дослідження:* бінарний генетичний алгоритм.

Питання для опрацювання

1. Бінарний ГА [13].

1. Задання до практичної роботи

1.1. Вивчити теоретичний матеріал.

1.2. Послідовно виконати такі завдання до практичної роботи «Розробка бінарного ГА розв'язання задачі визначення точки оптимуму заданої функції (у вигляді m- та ру-файлів)»:

А. Спроектувати бінарний ГА.

Б. Перевірити бінарний ГА на здатність розв'язувати задачу визначення точки оптимуму

2. Контрольні запитання та завдання

1. Опишіть алгоритм функціонування бінарного ГА.
2. Опишіть алгоритм функціонування неперервно-визначеного ГА.

Задачі для самостійного розв'язання (варіанти завдань):

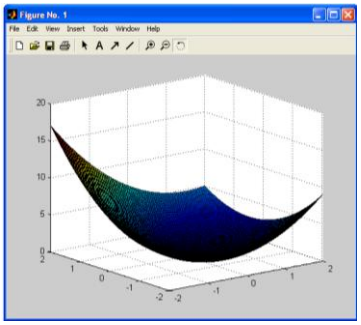
Реалізувати бінарний (неперервно-визначений) ГА та визначити точку оптимуму заданої функції:

№	Функція від $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, $-2 \le x_1 \le 2, -2 \le x_2 \le 2$
1	$\mathrm{F}(\mathbf{x}) = abs\left(x_1\right) - \cos\left(\frac{\pi}{3} x_2\right)$
2	$\mathrm{F}(\mathbf{x}) = abs\left(x_1\right) - \sin\left(\frac{\pi}{2} x_2\right)$
3	$\mathrm{F}(\mathbf{x}) = (x_2 - x_1)^2 + x_1 x_2 - x_1 - 2x_2 + 1 = \frac{1}{2}\mathbf{x}^{\mathrm{T}} \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \mathbf{x} + \begin{bmatrix} -1 & -2 \end{bmatrix} \mathbf{x} + 1$
4	$\mathrm{F}(\mathbf{x}) = abs\left(0,3x_1\right) + \sin\left(\frac{\pi}{2} x_2\right)$
5	$\mathrm{F}(\mathbf{x}) = abs\left(0,3x_1\right) - 10\cos\left(\frac{\pi}{12} x_2\right)$
6	$\mathrm{F}(\mathbf{x}) = -\left(-\left(x_1^2\right) + \cos\left(\frac{\pi}{6} x_2\right)\right)$

7	$\mathrm{F}(\mathbf{x}) = x_1 \cdot \sin\left(4x_1\right) + 1{,}1x_2 \cdot \sin\left(2x_2\right)$
8	$\mathrm{F}(\mathbf{x}) = x_2 \cdot \sin\left(4x_1\right) + 1{,}1x_1 \cdot \sin\left(2x_2\right)$
9	$\mathrm{F}(\mathbf{x}) = (x_2 - x_1)^2 + x_1x_2 - x_1 + x_2 + 1 = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}\mathbf{x} + \begin{bmatrix} -1 & 1 \end{bmatrix}\mathbf{x} + 1$
10	$\mathrm{F}(\mathbf{x}) = (x_2 - x_1)^2 + 2x_1x_2 + 2x_1 - x_2 + 3 = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}\mathbf{x} + \begin{bmatrix} 2 & -1 \end{bmatrix}\mathbf{x} + 3$
11	$\mathrm{F}(\mathbf{x}) = -\cos\left(\sqrt{\left(\left(x_1^2 + x_2^2\right)^2 - 0{,}5/\left(1 + 0{,}1\left(x_1^2 + x_2^2\right)\right)\right)}\right)$
12	$\mathrm{F}(\mathbf{x}) = 0{,}5 + \sin\left(\sqrt{\left(\left(x_1^2 + x_2^2\right)^2 - 0{,}5/\left(1 + 0{,}1\left(x_1^2 + x_2^2\right)\right)\right)}\right)$
13	$a = x_1^2 + x_2^2;\ b = \left(\left(x_1 + 0{,}5\right)^2 + x_2^2\right)^{0{,}1}$ $\mathrm{F}(\mathbf{x}) = a^{0{,}25} \cdot \sin^2\left(30b\right) + \mathrm{abs}(x_1) + \mathrm{abs}(x_2)$
14	$\mathrm{F}(\mathbf{x}) = \frac{1}{2}\left(10x_1^2 + 4x_2^2 - 2\sqrt{10}x_1x_2\right) - 2x_1 - x_2 + 1$
15	$\mathrm{F}(\mathbf{x}) = -\exp\left(0{,}2\sqrt{\left(x_1 - 1\right)^2 + \left(x_2 - 1\right)^2 + \left(\cos\left(2x_1\right) + \sin\left(2x_2\right)\right)}\right)$
16	$\mathrm{F}(\mathbf{x}) = x_1 \cdot \sin\left(\sqrt{abs\left(x_1 - \left(x_2 + 9\right)\right)}\right) - \left(x_2 + 9\right) \cdot \sin\left(\sqrt{abs\left(x_2 + 0{,}5x_1 + 9\right)}\right)$
17	$\mathrm{F}(\mathbf{x}) = (x_2 - 3x_1)^2 + 6x_1x_2 - x_1 + x_2 + 1 = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\begin{bmatrix} 18 & 0 \\ 0 & 2 \end{bmatrix}\mathbf{x} + \begin{bmatrix} -1 & 1 \end{bmatrix}\mathbf{x} + 1$
18	$\mathrm{F}(\mathbf{x}) = (x_2 - x_1)^2 + 2x_1x_2 + x_1 - x_2 + 1 = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}\mathbf{x} + \begin{bmatrix} 1 & -1 \end{bmatrix}\mathbf{x} + 1$
19	$\mathrm{F}(\mathbf{x}) = (x_2 - 2x_1)^2 + 4x_1x_2 + x_1 - 2x_2 + 1 = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\begin{bmatrix} 8 & -2 \\ -2 & 2 \end{bmatrix}\mathbf{x} + \begin{bmatrix} 1 & -2 \end{bmatrix}\mathbf{x} + 1$

Побудова графіка 3-D функції

```
[X,Y]=meshgrid([-2:0.05:2]);
Z=(Y-X).^2+X.*Y-X+Y+1;
surf (X,Y,Z)
```



```
function f=testfunction(x,funnum)
if funnum==1 %F1
f=abs(x(:,1))+cos(x(:,2));
elseif funnum==2 %F2
```

```
f=abs(x(:,1))+sin(x(:,2));end
```

Аудиторна робота

Приклад 22.1. Реалізація бінарного ГА в середовищі МатЛаб

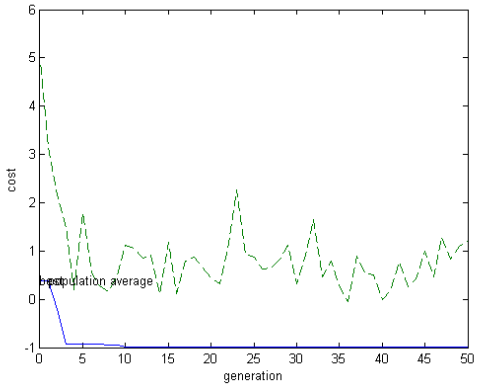
```
clc; clear all;
ff='testfunction'; % objective function
npar=2; % number of optimization variables
funnum=1;
%_____________________________________________________________
% Критерій зупинки
maxit=50; % max number of iterations
mincost=-9999999; % minimum cost
popsize=16; % set population size
mutrate=.15; % set mutation rate
selection=0.5; % fraction of population
% kept
nbits=8; % number of bits in each
% parameter
Nt=nbits*npar; % total number of bits
% in a chormosome
keep=floor(selection*popsize); % #population members
% that survive
%_____________________________________________________________
% Формування початкової популяції
iga=0; % generation counter
% initialized
pop=round(rand(popsize,Nt)); % random population of
% 1s and 0s
par=gadecode(pop,0,10,nbits); % convert binary to
% continuous values
cost=feval(ff,par,funnum) % calculates population
% cost using ff
[cost,ind]=sort(cost); % min cost in element 1
par=par(ind,:);pop=pop(ind,:); % sorts population with
% lowest cost first
minc(1)=min(cost); % minc contains min of
% population
meanc(1)=mean(cost); % meanc contains mean
% of population
while iga<maxit
iga=iga+1; % increments generation counter
M=ceil((popsize-keep)/2); % number of matings
prob=flipud([1:keep]'/sum([1:keep]));% weights
% chromosomes based
% upon position in
% list
odds=[0 cumsum(prob(1:keep))']; % probability
pick1=rand(1,M); % mate #1
pick2=rand(1,M); % mate #2
% ma and pa contain the indicies of the chromosomes
% that will mate
```

```
ic=1;
while ic<=M
for id=2:keep+1
if pick1(ic)<=odds(id) & pick1(ic)>odds(id-1)
ma(ic)=id-1;
end % if
 if pick2(ic)<=odds(id) & pick2(ic)>odds(id-1)
  pa(ic)=id-1;
 end % if
end % id
ic=ic+1;
end % while
ix=1:2:keep; % index of mate #1
xp=ceil(rand(1,M)*(Nt-1)); % crossover point
pop(keep+ix,:)=[pop(ma,1:xp) pop(pa,xp+1:Nt)];
% first offspring
pop(keep+ix+1,:)=[pop(pa,1:xp) pop(ma,xp+1:Nt)];
% second offspring
%_____________________________________________________________
% Мутація
nmut=ceil((popsize-1)*Nt*mutrate); % total number of mutations
mrow=ceil(rand(1,nmut)*(popsize-1))+1; % row to mutate
mcol=ceil(rand(1,nmut)*Nt); % column to mutate
for ii=1:nmut
pop(mrow(ii),mcol(ii))=abs(pop(mrow(ii),mcol(ii))-1);
% toggles bits
end % ii
%_____________________________________________________________
% The population is re-evaluated for cost
par(2:popsize,:)=gadecode(pop(2:popsize,:),0,10,nbits);
% decode
cost(2:popsize)=feval(ff,par(2:popsize,:),funnum);
% decode
[cost,ind]=sort(cost);
par=par(ind,:); pop=pop(ind,:);
%_____________________________________________________________
% Do statistics for a single nonaveraging run
minc(iga+1)=min(cost);
meanc(iga+1)=mean(cost);
%_____________________________________________________________
% Stopping criteria
if iga>maxit | cost(1)<mincost
break
end
[iga cost(1)]
end %iga
day=clock;
disp(datestr(datenum(day(1),day(2),day(3),day(4),day(5),day(6)),0)
) % Data
disp(['optimized function is ' ff])
format short g
```

```
disp(['popsize = ' num2str(popsize) ' mutrate = ' num2str(mutrate)
' # par = ' num2str(npar)])
disp(['#generations='      num2str(iga)      '      best      cost='
num2str(cost(1))])
disp(['best solution'])

disp([num2str(par(1,npar))]) % disp([num2str(par(1,:))])
disp('binary genetic algorithm')
disp(['each parameter represented by ' num2str(nbits) ' bits'])
figure(24)
iters=0:length(minc)-1;
plot(iters,minc,iters,meanc,'--');
xlabel('generation');ylabel('cost');
text(0,minc(1),'best');text(1,minc(2),'population average')
```

Результат

```
13-Dec-2017 06:43:15
optimized function is testfunction
popsize = 16 mutrate = 0.15 # par = 19
#generations=50 best cost=-0.99992
best solution   6.7843
binary genetic algorithm
each parameter represented by 8 bits
```

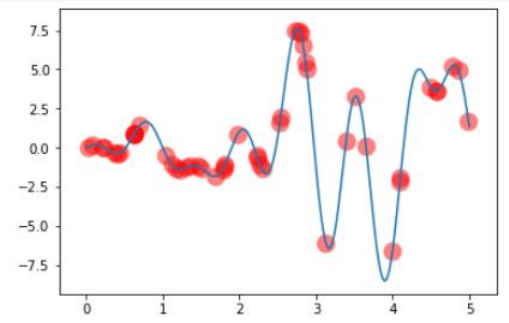


Приклад 22.2: python. Визначити максимум функції $y=\sin(3\cdot x)\cdot x+\cos(7\cdot x)\cdot x + \cos(9\cdot x)\cdot x$

Лістинг програми

```
import numpy as np
import matplotlib.pyplot as plt

DNA_SIZE = 10    # DNA length
POP_SIZE = 100   # population size
CROSS_RATE = 0.8 # mating probability (DNA crossover)
MUTATION_RATE = 0.003 # mutation probability
N_GENERATIONS = 200
X_BOUND = [0, 5] # x upper and lower bounds

def F(x):
 return np.sin(3*x)*x+np.cos(7*x)*x + np.cos(9*x)*x

def get_fitness(pred): return pred + 1e-3 - np.min(pred)

def translateDNA(pop): return pop.dot(2 ** np.arange(DNA_SIZE)[::-
1]) / float(2**DNA_SIZE-1) * X_BOUND[1]
def select(pop, fitness):
```

```
 idx=np.random.choice(np.arange(POP_SIZE),size=POP_SIZE,replace=
True,p=fitness/fitness.sum())
 return pop[idx]

def crossover(parent, pop):
 if np.random.rand() < CROSS_RATE:
  i_ = np.random.randint(0, POP_SIZE, size=1)
  cross_points=np.random.randint(0,2,size=DNA_SIZE).astype
(np.bool)
  parent[cross_points]=pop[i_, cross_points]
 return parent
def mutate(child):
 for point in range(DNA_SIZE):
  if np.random.rand() < MUTATION_RATE:
   child[point] = 1 if child[point] == 0 else 0
 return child

pop = np.random.randint(2, size=(POP_SIZE, DNA_SIZE))
plt.ion(); x = np.linspace(0, 5, 200)
plt.plot(x, F(x))
for _ in range(N_GENERATIONS):
F_values = F(translateDNA(pop))
if 'sca' in globals(): sca.remove()
sca   =   plt.scatter(translateDNA(pop),   F_values,   s=200,   lw=0,
c='red', alpha=0.5); plt.pause(0.05)
fitness = get_fitness(F_values)
print("Most fitted DNA: ", pop[np.argmax(fitness), :])
pop = select(pop, fitness)
pop_copy = pop.copy()
for parent in pop:
 child = crossover(parent, pop_copy)
 child = mutate(child)
parent[:] = child; plt.ioff(); plt.show()
```



Теоретичний матеріал

1. Бінарний генетичний алгоритм. Генетичний алгоритм – це евристичний алгоритм пошуку, який використовують для вирішення завдань оптимізації та моделювання шляхом використання механізмів генетичного

успадкування і природного відбору. В основу ГА покладено модель біологічної еволюції і методи випадкового пошуку. При цьому еволюційний пошук - це послідовне перетворення однієї кінцевої множини проміжних рішень завдання в іншу. Генетичний алгоритм реалізує випадковий пошук, ефективно використовуючи інформацію, накопичену в процесі еволюції.

Основна ідея ГА полягає в створенні популяції особин, кожна з яких зображена у вигляді хромосоми (можливого вирішення заданої задачі оптимізації).

Для пошуку найкращих рішень необхідне значення цільової функції (або функції пристосованості), яке відповідає певній особині і показує наскільки добре вона (і відповідна їй хромосома) підходить для вирішення завдання.

Хромосома складається з кінцевого числа генів, представляючи генотип об'єкта (сукупність його спадкових ознак). Останній оцінюють використовуючи функцію пристосованості, в результаті чого з кожним генотипом асоціюють певне значення (пристосованість). Процес еволюційного пошуку виконується тільки на рівні генотипу. До популяції застосовують основні «генетичні оператори» (схрещування, мутація та ін.), результатом чого є отримання нових розв’язків. У процесі еволюційного пошуку діє принцип «виживає найсильніший»: популяція постійно оновлюється за допомогою генерації нових особин і знищення старих, кожна нова популяція стає краще і залежить тільки від попередньої.

Компоненти бінарного ГА. Розглянемо процедуру реалізації бінарного ГА. Алгоритм розпочинається з визначення параметрів оптимізації, функції мети (ЦФ) та її значень, а закінчується перевіркою на збіжність. Блок-схема бінарного ГА зображена на рис. 23.1. Кожен блок детально розглянуто далі.

1. Вибір параметрів (змінних) і функції мети. Функція мети генерує вихідний сигнал на основі множини вхідних змінних (хромосом). Потрібно визначити значення цільової функції (ЦФ) і які змінні пов'язані з нею.

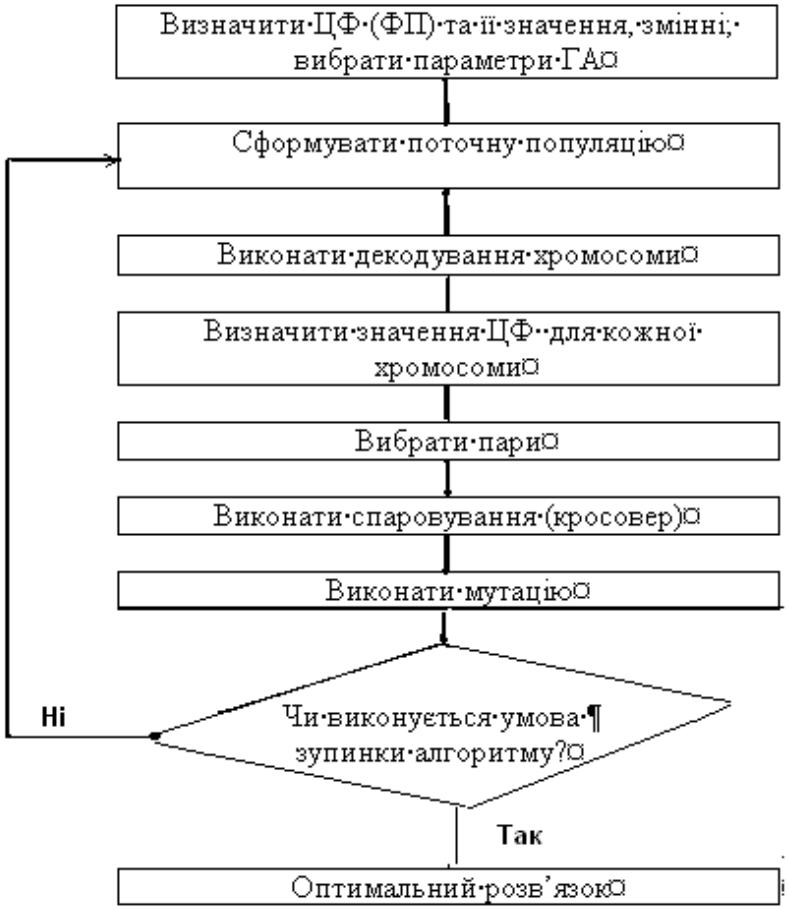


Рис. 22.1. Блок-схема бінарного ГА

Алгоритм розпочинається із визначення хромосом або масиву значень змінних, які повинні бути оптимізовані.

Нехай хромосома має *Nvar* змінних (маємо *Nvar*-вимірну задачу оптимізації): змінні p_1, …, p_{Nvar}. Хромосоми записують як вектор–рядок елементів N_{var}: *chromosome*=[p_1 … p_{Nvar}]. Наприклад, при пошуку максимальних точок (піків) на топографічній карті потрібна ЦФ з вхідними змінними довгота *x* и широта *y*: *chromosome*=[*x*, *y*], де N_{var}=2.

Кожній хромосомі відповідає значення ЦФ (придатність), отримане шляхом обчислення ЦФ $f(p_1, \ldots, p_{Nvar})$: $cost = f(chromosome) = f(p_1, \ldots, p_{Nvar})$.

В деяких випадках ЦФ задається аналітично, наприклад:

$f(w, x, y, z) = 2x + 3y + 100000z + 9876w$, де $x, y, z, w \in [1, 10]$.

На змінні можуть бути накладені обмеження вигляду «>», «<», «≥» і «≤». Коли змінна виходить за межі своєї області значень, то її встановлюють рівною

значенню межі (наприклад, якщо x змінюється в інтервалі [0, 10], а алгоритм ГА отримує нащадка $x = 11$, то x буде переведений до значення 10).

2. Кодування і декодування змінних. Оскільки значення змінної подають в двійковому вигляді, повинен бути спосіб перетворення неперервних величин у двійкові і навпаки. Цим способом є квантування прикладів неперервного ряду значень та їх класифікація на діапазони, що не перетинаються. Унікальне дискретне значення присвоюють кожному піддіапазону. Різниця між фактичним значенням функції і рівнем квантування відома як похибка квантування. Квантування розпочинає вибірку функції і розміщення зразків на рівних рівнях квантування (рис. 22.3), тут кожна хромосома відповідає значенням ЦФ: низький, середній або високий на рівні квантування функції (зазвичай параметр має середнє значення рівня квантування) і розміщення прикладів на однакових рівнях квантування. Будь-яке значення, яке входить в один із рівнів, встановлюють рівним середині, максимальному або мінімальному значенню цього рівня.

У загальному випадку встановлення значення в середині значення рівня квантування є найкращим, тому що найменша похибка можлива на половині рівня.

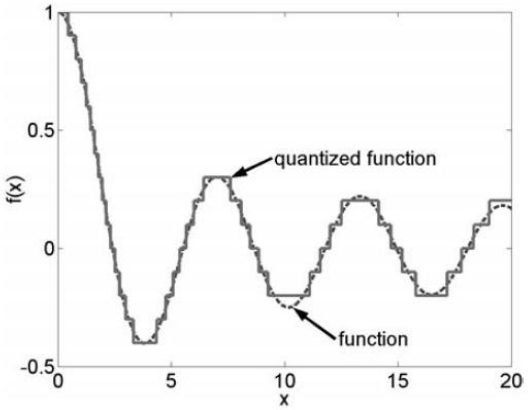
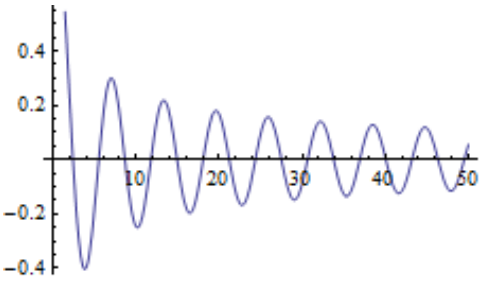


Рис. 22.2. Приклад квантування функції Бесселя J0(x)



	variable values				
	0.55	0.11	0.95	0.63	
111			•		0.9375
110					0.8125
101				•	0.6875
100	•				0.5625
011					0.4375
010					0.3125
001					0.1875
000		•			0.0625
	0.625	0.125	1.000	0.750	quantized hi
	0.500	0.000	0.875	0.625	quantized lo
	0.5625	0.0625	0.9375	0.6875	quantized mid
	100	000	111	101	chromosome

1.000
0.875
0.750
0.625
0.500
0.375
0.250
0.125
0.000

$$\text{Out(x)}= 1 - \frac{x^2}{4} + \frac{x^4}{64} - \frac{x^6}{2304} + \frac{x^8}{147\,456} - \frac{x^{10}}{14\,745\,600} + O[x]^{11}$$

Рис. 22.3. Графіки з чотирма неперервними значеннями змінних і рівнями квантування. Відповідний ген або хромосома вказує на рівень квантування, де значення змінної зменшується

Округлення значень на мінімальні або максимальні значення рівня дозволяє максимізувати похибку, що дорівнює рівням квантування. Математичні формули для двійкового кодування і декодування n-ої змінної p_n визначаються таким чином:

1) для кодування:

$$p_{norm} = \frac{p_n - p_{lo}}{p_{hi} - p_{lo}}; \quad gene[m] = \text{round}\left\{ p_{norm} - 2^{-m} - \sum_{p=1}^{m-1} gene[p] \cdot 2^{-p} \right\};$$

2) для декодування: $p_{quant} = \sum_{m=1}^{N_{gene}} gene[m] \cdot 2^{-m} + 2^{-(M+1)};$

$$q_n = p_{quant}\left(p_{hi} - p_{lo}\right) + p_{lo}, \qquad (22.1)$$

де p_{norm} – нормована змінна: $0 \le p_{norm} \le 1$; p_{lo} і p_{hi} – мінімальне і максимальне значення змінної; $gene\ [m]$ – двійкове зображення p_n; $round\ \{\}$ – округлення до найближчого цілого; p_{quant} – зображення значення квантування p_{norm}; q_n – зображення значення квантування p_n.

Двійковий ГА працює з бітами. Змінна x приймає значення, зображене рядком бітів, довжиною N_{gene}. Якщо $N_{gene} = 2$ і x має значення, яке визначається інтервалом $1 \le x \le 4$, то ген має $2^{Ngene} = 4$ можливих значень змінних.

Алгоритм працює з двійковим кодом, але ЦФ (функція придатності) часто вимагає неперервних змінних. Кожного разу, коли обчислюють значення ЦФ, хромосоми повинні бути спочатку декодовані з використанням (22.1). Наведемо приклад бінарної хромосоми, яка має N_{gene} закодованих змінних з N_{gene}= 10 біт:

$$chromosome = \left[\underbrace{1111001001}_{gene_1}\underbrace{0011011111}_{gene_2} \ldots \underbrace{0000101001}_{gene_{Nvar}}\right].$$

3. Популяція. Алгоритм розпочинається з групи хромосом (популяції), яка має N_{pop} хромосом і має вигляд матриці **pop** розмірності $N_{POP}\times N_{\text{bits}}$. Цю матрицю заповнюють випадковими одиницями і нулями (формують з використанням команди pop=round(rand(N_{pop}, N_{bits})), де функція rand(N_{POP}, N_{bits}) генерує матрицю $N_{POP}\times N_{bits}$ рівномірно розподілених випадкових чисел від «0» до «1». Функція round округлює число до найближчого цілого (в даному випадку це або «0», або «1»). Кожен рядок в **pop** зображує хромосому, якій відповідає значення ЦФ. Приклад початкової популяції та її цільові значення для N_{POP}=8 випадкових хромосом зображує табл. 22.1.

Таблиця 22.1

Приклад початкової популяції і відповідні їй ЦФ

Chromosome	Cost
00101111000110	–12359
11100101100100	–11872
00110010001100	–13477
00101111001000	–12363
11001111111011	–11631
01000101111011	–12097
11101100000001	–12588
01001101110011	–11860

4. Природний відбір. Розглянемо два підходи до природного відбору.

Рейтинговий підхід. Під час природного відбору при кожній генерації популяції (або ітерації алгоритму) найбільш пристосовані особини виживають, непристосовані – не виживають.

Хромосоми з великими значеннями ЦФ не приймають до уваги (розглядаємо задачу мінімізації).

З N_{POP} хромосом у популяції тільки хромосоми, які відповідають найменшим значенням ЦФ (верхня частина покоління N_{keep}) виживають для подальшого спарювання, іншу частину ($N_{POP} - N_{keep}$) видаляють, звільняючи місце для нових нащадків. Рішення, скільки хромосом зберігати, є довільним. Умова, що кілька хромосом доживуть до наступного покоління, обмежує кількість доступних генів у нащадках. Наявність великої кількості хромосом дозволяє поганим хромосомами внести свій вклад в ознаки наступної популяції.

Зазвичай в процесі природного відбору зберігають 50% (X_{rate}=0.5) популяції. У нашому прикладі, N_{POP}=8, маємо 50% відбір (тобто X_{rate}= 0.5), отже N_{keep}=4. Результат природного відбору наведено в табл. 22.2: хромосоми табл. 22.1 були спочатку відсортовані за значенням ЦФ, чотири з них із найменшими значеннями ЦФ переходять до наступного покоління і стають потенційними батьками.

Граничний підхід: всі хромосоми, значення ЦФ яких не перевищує деякий поріг, виживають. Ця межа дозволяє хромосомам розвиватися (як батьки) з метою народження нащадків (в іншому випадку, потрібно призвести ціле покоління, щоб визначити хромосоми, які відповідають певним вимогам).

Таблиця 22.2

Хромосоми, які вижили після 50% -го відбору

Chromosome	Cost
00110010001100	−13477
11101100000001	−12588
00101111001000	−12363
00101111000110	−12359

$$N_{keep} = X_{rate} N_{pop}$$

Спочатку (в першому поколінні) всього кілька хромосом можуть вижити, однак у пізніших поколіннях більшість хромосом виживе.

5. Вибір батьків. Прийшов час для кросовера (спарювання). Дві хромосоми обрані з пулу спарювання N_{keep} хромосом для створення двох нових нащадків. Парування відбувається в шлюбній частині популяції поки не сформуються (народяться) N_{pop}–N_{keep} нащадків, які замінюють видалені хромосоми. Розглянемо різні **методи селекції** (відбору батьків), розпочинаючи з найпростіших.

1. *Парування від верху до низу*. Розпочати процес зверху списку і формувати пари хромосом по два за раз, поки остання пара хромосом N_{keep} не буде відібрана для спарювання. Таким чином, алгоритм з'єднує непарні рядки з парними: мати має номери рядків в матриці популяції ma = 1,3,5, ..., батько – номери pa = 2, 4,6, ... Такий підхід ідеально не моделює природу, але дуже простий для програмування.

2. *Випадкове спарювання*. Цей підхід для вибору хромосом батьків використовує однорідний генератор випадкових чисел. Номери рядків батьків визначаються за формулою:

$$ma=ceil(N_{keep}\cdot rand(1, N_{keep})); \; pa=ceil(N_{keep}\cdot rand(1, N_{keep})),$$

де ceil округлює значення до найближчого більшого цілого.

3. *Виважене випадкове спарювання.* Хромосомами з шлюбного пулу (басейну рішень) відповідає ймовірність, яка обернено пропорційна значенню ЦФ. Хромосоми з найменшими значеннями ЦФ мають найбільшу ймовірність спарювання, а хромосоми з найбільшим значенням ЦФ мають найнижчу ймовірність спарювання. Випадкове число визначає, яка хромосома обрана. Цей метод часто називають як зважування за допомогою колеса рулетки. Існують дві методики: зважування за рангом і за вартістю (значенням ЦФ).

Зважування за рангом. Цей підхід знаходить ймовірність з рангу *n*–ої хромосоми: $P_n = \frac{N_{keep} - n + 1}{\sum_{n=1}^{N_{keep}} n} = \frac{4 - n + 1}{1 + 2 + 3 + 4} = \frac{5 - n}{10}$. Результат для популяції розміром N_{keep}=4 хромосом зображує табл. 22.3.

Таблиця 22.3

Ранг зважування

n	Chromosome	P_n	$\sum_{i=1}^{n} P_i$
1	00110010001100	0,4	0,4
2	11101100000001	0,3	0,7
3	00101111001000	0,2	0,9
4	00101111000110	0,1	1,0

При виборі хромосоми використовують сукупні ймовірності, перераховані в четвертому стовпчику. Генерується випадкове число між нулем і одиницею. Починаючи з верхньої частини списку (з «басейну» рішень) для спарювання вибирають першу хромосому з ймовірністю, яка більше випадкового числа. Наприклад, якщо випадкове число r=0,77, то $0, < r \le 0,7$, так що обрана хромосома *chromosome2*.

Якщо хромосома парується сама з собою, то існує кілька альтернатив: 1) залишити її (що означає: в наступному поколінні є три однакові хромосоми); 2) випадково вибрати іншу хромосому; 3) вибрати іншу хромосому, використовуючи метод зважування. Запрограмувати зважування за рангом трохи складніше, ніж спаровування від верху до низу. Невеликі популяції мають високу ймовірність вибору однієї і тієї ж хромосоми. Ймовірності повинні бути обчислені тільки один раз. Ми схильні використовувати зважування за рангом, тому що значення ймовірностей не змінюються для кожного покоління.

Зважування за значенням ЦФ. Можливість вибору обчислюється на підставі ЦФ хромосоми, а не її рангу в популяції. Для кожної хромосоми розраховують нормалізоване значення шляхом обчислення найменшого

значення відкинутих хромосом $C_{Nkeep\text{-}1}$ від значень всіх хромосом в пулі спарювання: $C_n=C_n-C_{Nkeep\text{-}1}$ Віднімання $C_{Nkeep\text{-}1}$ забезпечує той факт, що всі значення ЦФ є від'ємними. Таблиця 22.4 перераховує нормалізовані значення ЦФ за умови, що $C_{Nkeep\text{-}1}$=– 12097. Значення p_n розраховують за формулою: $P_n = \left| \frac{C_n}{\sum_{m}^{N_{keep}} C_m} \right|$. Цей підхід має такі тенденції: 1) якщо є великий розкид в значеннях ЦФ між верхньою і нижньою хромосомами, то зважувати верхню хромосому великим значенням; 2) якщо всі хромосоми мають приблизно однакові значення ЦФ, то зважувати хромосоми рівномірно.

Існують проблеми, якщо хромосому вибирають, щоб парувати її саму з собою. Ймовірності повинні бути розраховані для кожного покоління.

Таблиця 22.4

Значення ЦФ із зважуванням

n	Chromosome	$C_n=c_n-c_{Nkeep+1}$	P_n	$\sum_{i=1}^{n} P_i$
1	00110010001100	–13477+12097=–1380	0,575	0,575
2	11101100000001	–12588+12097=–491	0,205	0,780
3	00101111001000	–12363+12097=–266	0,111	0,891
4	00101111000110	–12359+12097=–262	0,109	1,000

Відбірковий турнір. Це є підхід, який імітує змагання при спарюванні в природі. Він полягає в тому, щоб випадково вибрати невелику підмножину хромосом (дві або три) з пулу спарювання, і хромосоми з найменшим значенням ЦФ в цій підмножині стають батьками. Турнір повторюється при кожній потребі у батьках.

Встановлення значень межі і відбірковий турнір формують гарну пару (бо популяцію не треба сортувати). Відбірковий турнір найкраще працює для великих розмірів популяції, тому що сортування популяції забирає багато часу у популяцій великих груп.

Кожна схема вибору батьків призводить до різних наборів батьків. Таким чином, склад наступного покоління відрізняється для кожної схеми селекції.

Колесо рулетки і відбірковий турнір є стандартними для більшості ГА. Важко сказати, яка схема зважування працює найкраще. В нашому прикладі ми досліджуємо процедуру відбору на основі зважування за рангом батьків.

6. Парування. Парування є створенням одного або декількох нащадків від батьків, відібраних в процесі формування пар. Завдяки своїй генетичній структурі популяція обмежена складом її елементів. Найбільш поширена форма спарювання має на увазі двох батьків, які формують двох нащадків (рис. 22.4). Випадковим чином вибирають точку кроссовера між першим і останнім бітами хромосом батьків: *parent*1 і *parent*2.

Хромосома *parent*1 передає свій двійковий код зліва від точки кросовера нащадку *offspring*1.

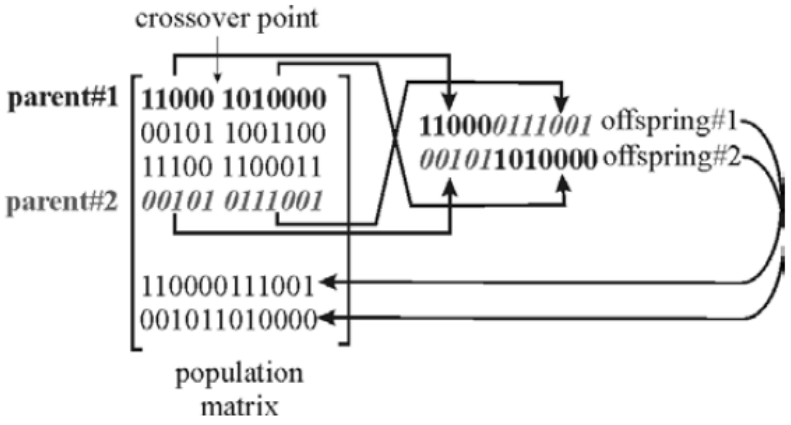


Рис. 22.4. Парування двох батьків для відтворення двох нащадків offspring (нащадків розміщують в популяцію)

Подібним чином, *parent*2 передає свій двійковий код зліва від точки кросовера нащадку *offspring*2. Потім двійковий код праворуч від точки кросовера з *parent*1 йде до *offspring*2, а *parent*2 передає свій код праворуч нащадку *offspring*1. Отже, нащадки містять частини двійкових кодів обох батьків. Батьки підготували в цілому ($N_{pop} - N_{\text{keep}}$) нащадків, так що загальна кількість хромосом популяції стала N_{pop}. У табл. 22.7 зображені батьки і нащадки.

Таблиця 22.7

Процес спаровування одно–точкового кросовера

Chromosome	Family	Binary String
3	ma(1)	*00101111001000*
2	pa(1)	11101100000001
5	$offspring_1$	*00101*100000001
6	$offspring_2$	11101*111001000*
3	ma(2)	*00101111001000*
4	pa(2)	00101111000110
7	$offspring_3$	*00101111000*110
8	$offspring_4$	00101111001*000*

Перша множина хромосом батьків {3, 2} має точку кросовера між бітами 5 і 6, друга множина {3, 4} – точку між бітами 10 і 11. Цей процес відомий як одно–точковий кросовер (існують й інші види цих операторів).

7. Мутація. Випадкові мутації змінюють певний відсоток бітів у списку хромосом. Мутація може внести в генофонд відмінні ознаки, які були відсутні в початковій популяції, і перешкоджає занадто швидкій збіжності ГА.

Одноточкова мутація змінює 1 на 0 і навпаки. Точки мутації вибираються випадковим чином із загальної кількості бітів $N_{pop} \times N_{bits}$ в матриці популяції. Збільшення числа мутацій збільшує свободу алгоритму пошуку в поточній області простору змінних і відволікає алгоритм від збіжності до популярного рішення. Мутація не виконується на останній ітерації.

Чи допускається мутація на кращих рішеннях? – Взагалі ні. Вони визначаються як *елітні рішення*, призначені для поширення без змін. Такий елітизм дуже поширений в ГА. Навіщо викидати ідеальну відповідь?

В нашому прикладі мутують 20% особин популяції (m=0,20), за винятком кращої хромосоми. Таким чином, генератор випадкових чисел створює сім пар випадкових чисел, які відповідають рядкам і стовпчикам бітів, що мутують. У цьому випадку кількість мутацій визначається за формулою:

```
#mutations = m*(Npop - 1)*Nbits = 0.2*7*14 = 19.6 ≈ 20
```

Комп'ютерний код визначення рядка і стовпчика бітів, які мутують:

```
nmut=ceil((Npop - 1)*Nbits m);
mrow=ceil(rand(1, m)*(Npop - 1))+1;
mcol=ceil(rand(1, m)*Nbits);
```

```
pop(mrow,mcol)=abs(pop(mrow,mcol)-1);
```

Наступні пари були відібрані випадковим чином:
mrow=[5 7 6 3 6 6 8 4 6 7 3 4 7 4 8 6 6 4 6 7]
mcol= [6 12 5 11 13 5 5 6 4 11 10 6 13 3 4 11 5 14 10 5]

Першою випадковою парою є (5, 6). Таким чином, біт в рядку 5 і стовпчику 6 матриці популяції мутує з 1 на 0:

$$00101100000001 \Rightarrow 00101\boldsymbol{0}00000001$$

Мутації відбуваються ще 19 раз. Мутовані біти в табл. 22.8 виділені курсивом. Перша хромосома не мутувала через елітизм. Тільки 18 біт мутували в табл. 22.8 замість 20 (пара рядок–стовпчик (6, 5) були випадково відібрані три рази, тому один і той же біт змінювався з «1» на «0», потім з «0» на «1» і, нарешті, з «1» на «0»). Хромосоми в кінці першого покоління зображені на рис. 22.6.

8. Наступні покоління. Після мутації розраховують значення ЦФ, пов'язані з нащадками і хромосомами, які мутують (третій стовпчик в табл. 22.8). Описаний процес повторюється. В табл. 22.9 зображена початкова популяція для формування 2-го покоління (4 нижні хромосоми рейтингу відкидаються і замінюються нащадками від чотирьох найкращих батьків).

Таблиця 22.8

Процес мутації

Популяція до мутації	*Популяція після мутації*	*New Cost*
00110010001100	00110010001100	-13477
11101100000001	11101100000001	-12588
00101111001000	00101111010000	-12415
00101111000110	00001011000111	-13482
00101100000001	00101000000001	-13171
11101111001000	11110111010010	-12146
00101111000110	00100111001000	-12716
00101111001000	00110111001000	-12103

Ще 20 випадкових біт вибирають для мутації з нижніх 7 хромосом і т. д.

Таблиця 22.9

Нова ранжована популяція на початку 2-го покоління

Хромосома	*Cost*
00001011000111	-13482
00110010001100	-13477
00101000000001	-13171
00100111001000	-12716
11101100000001	-12588
00101111010000	-12415
11110111010010	-12146
00110111001000	-12103

9. Збіжність. Число поколінь залежить від того, чи досягнуто прийнятне рішення або перевищено задану кількість ітерацій. Якби не було мутацій, то через деякий час всі хромосоми і пов'язані з ними значення ЦФ стали б однаковими. У цей момент алгоритм повинен бути зупинений.

Деякі ГА відстежують статистику популяції у вигляді середніх і мінімальних значень ЦФ популяції.

23. Самоорганизація і самоналаштування нечітких моделей методами пошуку

Мета роботи: ознайомитися з деякими алгоритмами самоорганизації і самоналаштування нечітких моделей методами пошуку. *Об'єкт дослідження:* алгоритми самоорганизації і самоналаштування нечітких моделей методами пошуку.

Питання для опрацювання

1. Самоорганизація і самоналаштування нечітких моделей методами пошуку: налаштування параметрів нечіткої моделі (параметрів вхідних і вихідних ФН); одночасне налаштування і самоорганізація параметрів нечіткої моделі [7].

1. Завдання до практичної роботи

1. Вивчити теоретичний матеріал.

2. Послідовно виконати такі завдання до практичної роботи «Розробка системи розв'язання задачі алгоритмами самоорганизації і самоналаштування нечітких моделей методами пошуку (у вигляді m- та ру-файлів)»: виконати налаштування параметрів нечіткої моделі (параметрів вхідних і вихідних ФН); перевірити нечітку модель на здатність розв'язувати задачу апроксимації.

2. Контрольні запитання та завдання

1. Опишіть алгоритм розв'язання задачі налаштування параметрів нечіткої моделі (параметрів вхідних і вихідних ФН).

2. Опишіть алгоритм розв'язання задачі одночасного налаштування і самоорганізації параметрів нечіткої моделі.

Задачі для самостійного розв'язання (варіанти завдань): комп'ютерний практикум № 4

Реалізувати систему розв'язання задачі апроксимації на основі налаштування параметрів нечіткої моделі (параметрів вхідних і вихідних ФН).

Теоретичні відомості

Самоорганизація і самоналаштування нечітких моделей методами пошуку [7].

1. Налаштування параметрів нечіткої моделі (параметрів вхідних і вихідних ФН). Розглянемо задачу налаштування параметрів нечіткої моделі (параметрів вхідних і вихідних ФН). Генетичні алгоритми дозволяють виконувати налаштування більшості нечітких моделей, однак їх використання пов'язане зі значними часовими витратами. В контексті ГА область значень кожної змінної розглядається як простір, розбитий на скінчену кількість

інтервалів (рис. 23.1). Якщо заданий інтервал містить пік ФН, то він кодується за допомогою цифри «1», в іншому випадку – за допомогою цифри «0». Закодований рядок, який відповідає ФН однієї змінної, називають хромосомою (10110001011), а її елементи (0 або 1) називають генами. Розглянемо один із існуючих методів кодування.

Збільшення кількості генів, які формують хромосому (тобто числа інтервалів розбиття області значень), призводить до збільшення роздільної здатності пошуку оптимального рішення. Разом із тим, чим вищу роздільну здатність має розбиття, тим швидше зростає кількість потенційних рішень, що призводить до збільшення трудомісткості.

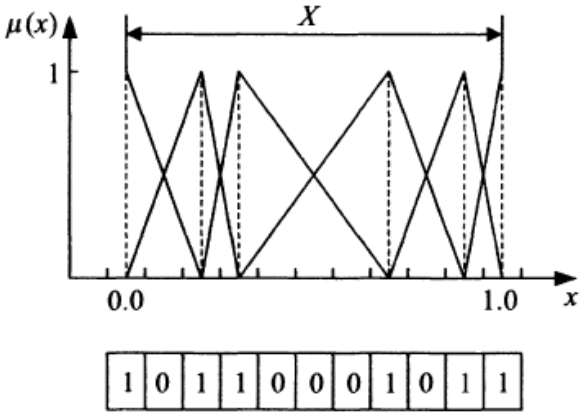


Рис. 23.1. Приклад коду ФН у вигляді генетичного рядка

У разі системи з багатьма входами генетичному кодуванню підлягають всі вхідні та вихідні параметри моделі (рис. 23.2).

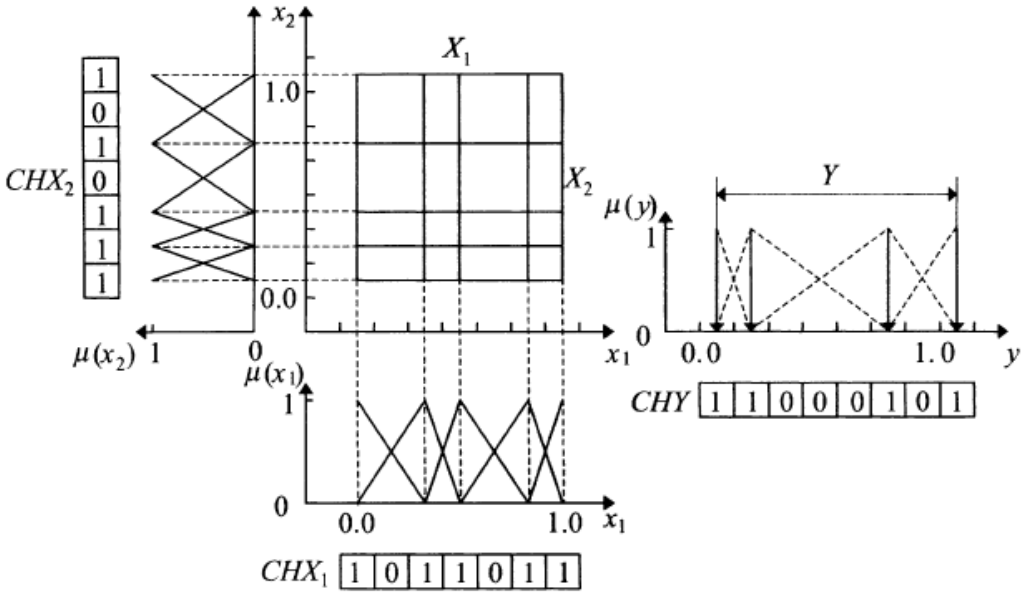


Рис. 23.2. Приклад генетичного зображення нечіткої моделі з двома входами, що задається за допомогою фіксованої кількості множин для кожної змінної та фіксованої бази правил

При фіксованому виборі кількості нечітких множин, які використовують для оцінки всіх вхідних і вихідних параметрів моделі, а також бази правил та інших елементів моделі, її точність буде залежати тільки від координат піків окремих ФН, тобто від розташування одиниць в хромосомах. Точність моделі, яку називаюь її функцією пристосованості (оцінки) D_i залежить від конкретного генетичного зображення нечіткої моделі R_i: $D_i = f(R_i)$.

Перед початком процесу пошуку випадковим або будь-яким іншим способом генерується початкова популяція $P_0 = \{R_1/D_1, \ldots, R_m/D_m\}$. З кожним поданням R_i пов'язана відповідна їй точність моделі D_i.

Зображення нечіткої моделі R_i, які містяться в початковій популяції, використовують для генерації нових зображень R^*_i (батьки формують нащадків). З цією метою застосовуються генетичні оператори, які здійснюють модифікацію генів у відповідних хромосомах (зміну позицій піків функцій належності). Основними генетичним операторами є мутація і кросинговер.

1. *Кросинговером* (рис. 23.3) називають процес обміну одного або більше генів між двома хромосомами (батьками) для отримання нових хромосом (нащадків).

2. *Мутацією* (рис. 23.4) називається процес утворення нащадків R^*_i заданого зображення нечіткої моделі R_i шляхом модифікації одного або декількох його генів («1» → «0» або «0» → «1»). Якщо процес налаштування обмежений виключно ФН, мутацію виконують таким чином, щоб кількість 1-генів (нечітких множин для відповідної змінної) залишалася постійною.

$(CHX_1)_1$: 0 1 0 1 1 0 $\quad$ $(CHX_1)^*_1$: 0 1 0 1 0 1

$(CHX_1)_2$: 1 0 0 1 0 1 $\quad$ $(CHX_1)^*_2$: 1 0 0 1 1 0

Рис. 23.4. Приклад використання оператора кросинговера: $(CHX_1)_1$, $(CHX_1)_2$ – батьки, $(CHX_1)^*_1$, $(CHX_1)^*_2$ – нащадки

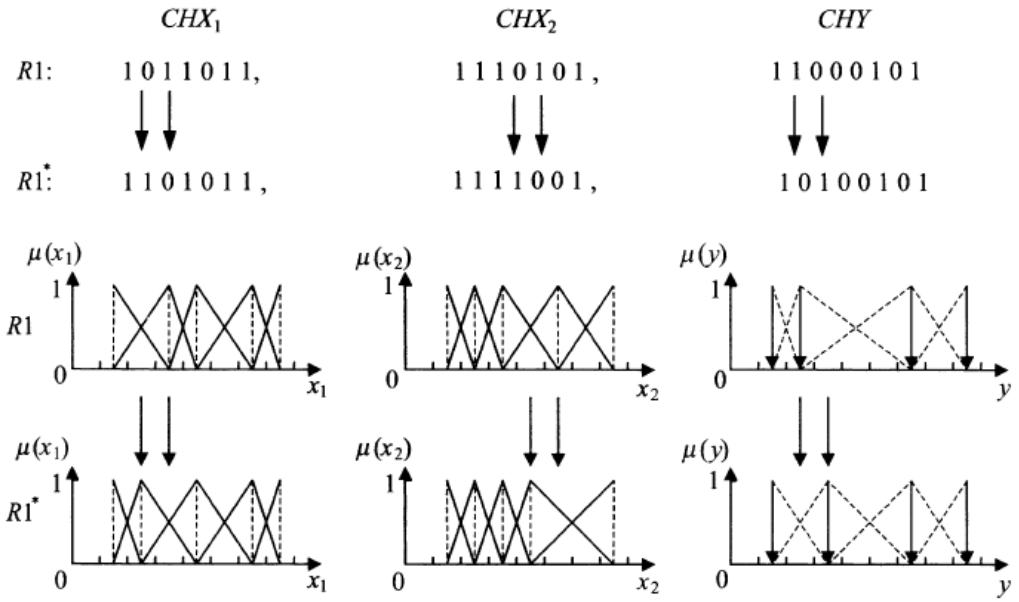


Рис. 23.4. Приклад застосування оператора мутації для формування нового зображення нечіткої моделі

Якщо налаштування зачіпає виключно ФН, кросинговер необхідно встановити так, щоб кількість «1» –генів в хромосомі кожного типу (CHX_1, CHX_2, CHY) залишалася незмінною. Кожному новому зображенню нечіткої моделі R^*_i ставиться у відповідність його ступінь пристосованості D^*_i, і вона вводиться в початкову популяцію, формуючи початок нової популяції P. Нову популяцію проглядають, і з неї виключають найгірші зображення моделі (ті, у яких ступінь пристосованості є мінімальною). До зображень, що залишилися, застосовують генетичні операції для утворення нових, більш перспективних нащадків. Цей процес повторюють, поки серед новостворених зображень моделі не буде знайдено таке зображення, ступінь пристосованості якого задовольняє вимогам точності моделі.

Таким чином, ГА включає в себе такі операції:

1) кодування завдання для отримання його генетичного зображення моделі R і визначення функції пристосованості $D(R)$ для кожного подання,

2) формування початкової популяції P_0,

3) розмноження та відбір отриманої популяції, поки не буде отримано зображення моделі R_0, яке задовольняє вимогам, пов'язаним зі ступенем пристосованості (наприклад, $D_0 \leq D_{min}$).

Існує багато методів вирішення завдань із застосуванням ГА. У наш час ГА формують науковий напрямок, який неперервно розвивається. Застосування ГА надає можливість пошуку оптимальних рішень в задачах підвищеної складності. Тому цей метод отримав велику популярність, і кількість його додатків постійно зростає. Разом з тим, метод має недолік, пов'язаний з необхідністю розбиття простору значень змінних моделі на скінчену кількість інтервалів, щоб забезпечити скінчене число біт в генетичному зображенні моделі. Завдяки дискретизації простору рішень не завжди вдається отримати такі ж гарні рішення, як і в разі застосування методів, що діють в неперервному просторі (наприклад, нейронечітких мереж). Проте, при правильному виборі роздільчої здатності дискретизації вказаний недолік практично не проявляється, тому ГА можна рекомендувати до застосування. Крім цього, додаткова перевага цього методу полягає в можливості його застосування при дослідженні впливу на точність моделі різних її складових (типів операторів «І» та «АБО», методів виведення і дефаззифікації, використовуваних типів ФН тощо).

Якщо в якості однієї зі змінних моделі ввести тип оператора «І», областю значень якого є множина, яка містить сім операторів t-норми: {I}={MIN, посилений добуток, оператор І. Лукасевича, добуток Ейнштейна, добуток Гамахера, алгебраїчний добуток, оператор І. Ягера}, то зображення моделі слід розширити, додавши семібітну хромосому вигляду CHAND = 0000010, позиція 1-гена в якій відповідає обраному оператору «І» з множини. В результаті зображення моделі має вигляд:

	CHX 1	CHX_2	CHY	CH AND
$R1$:	1011011,	1110101,	11000101,	0000010

Однак, додавання нових хромосом хоча і дозволяє знайти кращу модель, але вимагає значно більшого обсягу обчислень.

2. Одночасне налаштування і самоорганізація параметрів нечіткої моделі. Розглянемо задачу одночасного налаштування і самоорганізації параметрів нечіткої моделі. Термін «самоорганізація» означає визначення кількості і форми правил, вигляду функцій належності входів і виходів моделі, а також, по можливості, типів операторів «І», «АБО» і методу дефаззифікації. Пошук оптимальної структури і параметрів можна здійснювати методом проб і помилок, досліджуючи різні види структур. Разом з тим, при застосуванні даного методу багато залишається за межами нашої уваги, і процес пошуку оптимальної моделі може займати тривалий час, тому в подібних ситуаціях бажано мати попередню інформацію про систему, яка моделюється.

Метод ГА дозволяє виконувати оптимізацію найбільш складних структур моделей. Основною його перевагою є здатність знаходити глобальне оптимальне (або субоптимальное) рішення з урахуванням практично всіх можливих обмежень, які можна на нього накласти різними типами задач.

Часто ГА використовують як основу методу оптимізації нечітких моделей. Розглянемо метод генетичного пошуку оптимальної структури нечіткої моделі, розроблений Нелесом (Nelles, 1996 р.). Перевагою цього методу є його можливість побудови «економних» моделей, які містять малу кількість правил, що, в свою чергу, дозволяє ефективно застосовувати цей метод до систем з досить великою кількістю входів.

Для оптимізації завдання з використанням ГА це завдання необхідно зобразити (закодувати) у вигляді двійкового рядка, названого особиною (хромосомою або організмом). *Хромосомою* називають рядок закодованих параметрів завдання або елементів його структури, які підлягають оптимізації. Кожна хромосома є одним із можливих рішень задачі і, таким чином, відповідає точці багатовимірного простору пошуку. Елементи хромосом, які називають генами, зображують окремі елементи розв'язуваної задачі оптимізації. Множина

хромосом утворює популяцію. Розмір популяції, в якій проводиться пошук найкращої особини, визначається користувачем. Кожна хромосома, яка входить в популяцію, оцінюється за допомогою критерію, названого *функцією пристосованості* (ФП). Розглянемо послідовність кроків традиційного ГА.

Крок 1. **Ініціалізація алгоритму.**

Кодування завдання у вигляді генів і хромосом. Визначення ФП.

Визначення умови завершення алгоритму (мінімально допустимого значення ФП). Вибір початкової популяції хромосом.

Крок 2. **Оцінка ступеня пристосованості хромосом в популяції.**

Крок 3. **Перевірка умови завершення пошуку** і, при його виконанні, вибір хромосоми (рішення), яке задовольняє даній умові (кінець пошуку). В іншому випадку перехід до кроку 4.

Крок 4. **Селекція хромосом:** класифікація їх на «найкращі» і «найгірші» для відбору «кандидатів» на формування нових хромосом.

Крок 5. **Виконання генетичних операцій**. Відібрані хромосоми-«кандидати» використовують для формування нових хромосом за допомогою спеціальних генетичних операторів.

Крок 6. **Створення нової популяції**. Нова популяція підлягає оцінюванню – повернення до кроку 2.

У кожному новому поколінні, тобто на кожній ітерації алгоритму відбувається покращення і посилення популяції хромосом (рішень). Ймовірність того, що конкретна хромосома переживе селекцію, тобто стане елементом наступної популяції, пропорційна її ступеню пристосованості. Шляхом селекції здійснюється зсув популяції в просторі рішень у напрямку областей підвищення пристосованості. Найчастіше використовують генетичні оператори мутації і кросинговеру.

1. Операція *кросинговера* розбиває зображення рядків двох хромосом у випадково обраній позиції і виконує між ними обмін фрагментами, отриманими в результаті розтину.

2. Операція *мутації* змінює в зображенні рядків хромосоми значення одного з бітів на протилежне.

Ймовірність кросинговеру і мутації хромосом задається користувачем. За аналогією з природними умовами, ймовірність схрещування зазвичай близька до «1», а ймовірність мутації близька до «0» (наприклад, значення ймовірності кросинговера p_c=0,9, а ймовірність мутації p_m=0,2). Вибір ймовірності є непростим завданням, оскільки не всі біти (гени), які формують хромосому, мають однакову, з точки зору оптимізації, значимість. Наприклад, хромосома «10000» відповідає десятковому числу «16», і в результаті мутації першого гена буде отримана хромосома «00000», яка відповідає десятковому числу «0» (в даному випадку мутація привела до значних змін). Якщо мутації піддається останній ген хромосоми «00001», то в результаті також буде отримана хромосома «00000», що в десятковій системі відповідає зміні «1» на «0». Тому вважаємо за доцільне кожному гену ставити у відповідність своє значення мутації.

Основним завданням, що виникає в процесі пошуку оптимальної бази правил, є розробка відповідного методу кодування. На початку необхідно на основі попередньо визначених функцій належності визначити множину всіх можливих правил моделі.

Розглянемо систему з двома входами і одним виходом. Припустимо, що з кожним входом пов'язані три трикутні ФН, а з виходом – множина одноточкових ФН, при цьому кожному правилу відповідає свій одноточковий висновок. На рис. 23.5 зображено розбиття вхідного простору X, а також одноточкові множини B_j в просторі Y.

База правил може містити як елементарні, так і узагальнюючі правила, які є логічними комбінаціями елементарних правил. Прикладом елементарного правила є правило вигляду R_j: ЯКЩО (x_1 = A_{13}) І (x_2 = A_{22}), ТО (y = B_j).

Елементарне правило ставить у відповідність множинам A_{13} і A_{22} вхідного простору одноточкову множину B_j вихідного простору. Це означає, що

область вихідного простору навколо точки b_j, в якій знаходиться одноточкова множина B_j, відповідає області вхідного простору навколо точки P_6 (рис. 23.5).

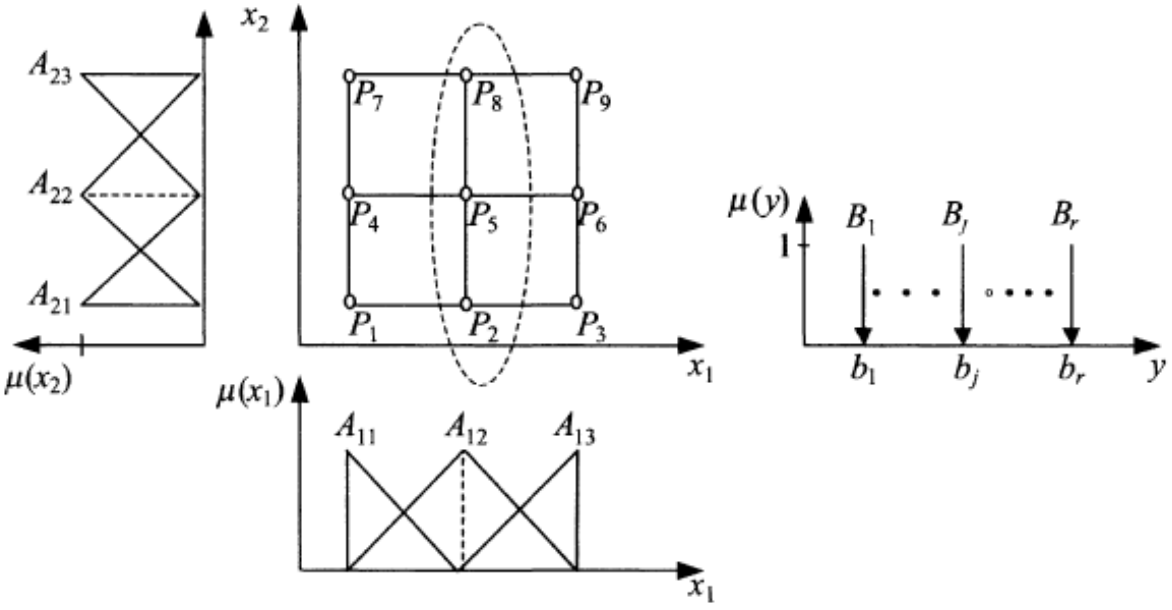


Рис. 23.5. Приклад розбиття вхідного простору на основі обраних ФН

Прикладом узагальнюючого правила є правило

R_j: ЯКЩО ($x_1 = A_{12}$), ТО ($y = B_j$), (23.1)

отримане на основі логічної комбінації таких трьох елементарних правил:

$R2$: ЯКЩО ($x_1 = A_{12}$) І ($x_2 = A_{21}$), ТО ($y = B_j$),

$R5$: ЯКЩО ($x_1 = A_{12}$) І ($x_2 = A_{22}$), ТО ($y = B_j$),

$R8$: ЯКЩО ($x_1 = A_{12}$) І ($x_2 = A_{23}$), ТО ($y = B_j$).

Ця логічна комбінація має вигляд:

«$R2$ АБО $R5$ АБО $R8$» =

= ЯКЩО ($x_1 = A_{12}$) І (($x_2 = A_{21}$) АБО ($x_2 = A_{22}$) АБО ($x_2 = A_{23}$)), ТО ($y = B_j$) =

= ЯКЩО ($x_1 = A_{12}$), ТО ($y = B_j$).

За допомогою одного узагальнюючого правила можна описати велику область вхідного простору, що включає околи точок P_2, P_5, P_8 (рис. 23.5). Оскільки для цього правила один і той самий висновок ($y = B_j$) відноситься до більшої за розміром області вхідного простору, ніж в разі елементарного правила, то точність моделі, що містить узагальнюючі правила, зазвичай нижча за точність моделі з елементарними правилами. З іншого боку, можливість

опису великої області на основі одного правила дозволяє скоротити кількість правил та отримати «економну» модель. Тим самим, застосування даного методу дозволяє уникнути явища, названого «прокляттям розмірності».

Оптимальна нечітка модель може містити елементарні та узагальнюючі правила. Тому до початку кодування задачі необхідно вказати множину всіх можливих правил R_j. У розглянутому прикладі подібна множина має такий вигляд:

R_1: ЯКЩО ($x_1 = A_{11}$), ТО ($y = B_1$)
R_2: ЯКЩО ($x_1 = A_{12}$), ТО ($y = B_2$)
R_3: ЯКЩО ($x_1 = A_{13}$), ТО ($y = B_3$)
R_4: ЯКЩО ($x_1 = A_{21}$), ТО ($y = B_4$)
R_5: ЯКЩО ($x_1 = A_{22}$), ТО ($y = B_5$)
R_6: ЯКЩО ($x_1 = A_{23}$), ТО ($y = B_6$)
R_7: ЯКЩО ($x_1 = A_{11}$) І ($x_2 = A_{21}$), ТО ($y = B_7$)
R_8: ЯКЩО ($x_1 = A_{11}$) І ($x_2 = A_{22}$), ТО ($y = B_8$)
R_9: ЯКЩО ($x_1 = A_{11}$) І ($x_2 = A_{23}$), ТО ($y = B_9$)
R_{10}: ЯКЩО ($x_1 = A_{12}$) І ($x_2 = A_{21}$), ТО ($y = B_{10}$)
R_{11}: ЯКЩО ($x_1 = A_{12}$) І ($x_2 = A_{22}$), ТО ($y = B_{11}$)
R_{12}: ЯКЩО ($x_1 = A_{12}$) І ($x_2 = A_{23}$), ТО ($y = B_{12}$)
R_{13}: ЯКЩО ($x_1 = A_{13}$) І ($x_2 = A_{21}$), ТО ($y = B_{13}$)
R_{14}: ЯКЩО ($x_1 = A_{13}$) І ($x_2 = A_{22}$), ТО ($y = B_{14}$)
R_{15}: ЯКЩО ($x_1 = A_{13}$) І ($x_2 = A_{23}$), ТО ($y = B_{15}$) (23.2)

Кожному елементу R_j множини можливих правил ставиться у відповідність певний ген в хромосомі (рис. 23.6).

R1	*R2*	*R3*	*R4*	*R5*	*R6*	*R7*	*R8*	*R9*	*R10*	*R11*	*R12*	*R13*	*R14*	*R15*
1	0	1	0	0	0	0	0	0	1	1	1	0	0	0

Рис. 23.6. Хромосома, яка відповідає одній з можливих структур БП

Позиції, зайняті 1-генами, задають одну конкретну структуру бази правил. Хромосомі, зображеній на рис. 23.6, відповідає така база правил

R_1: ЯКЩО ($x_1 = A_{11}$), ТО ($y = B_1$)
R_3: ЯКЩО ($x_1 = A_{13}$), ТО ($y = B_3$)
R_{10}: ЯКЩО ($x_1 = A_{12}$) І ($x_2 = A_{21}$), ТО ($y = B_{10}$)
R_{11}: ЯКЩО ($x_1 = A_{12}$) І ($x_2 = A_{22}$), ТО ($y = B_{11}$)
R_{12}: ЯКЩО ($x_1 = A_{12}$) І ($x_2 = A_{23}$), ТО ($y = B_{12}$)

Ця база покриває всю область значень входів. При використанні тільки

елементарних правил для покриття тієї ж області треба було б 3·3=9 правил (*R*7 ÷ *R*15), які входять у базу (23.2).

На етапі ініціалізації алгоритму визначають *функцію пристосованості*. Для оцінювання структури моделі доцільно використовувати функцію *F*, яка враховує кількість правил в базі *R*, точність моделі і виражається, наприклад, за допомогою абсолютної похибки *E* при використанні конкретної вибірки вимірювань, $F = \frac{1}{E + \alpha R}$, де $E = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - y_i^m\right|$, N – кількість елементів вибірки вимірювань, y - вихідне значення системи, y^m – значення на виході моделі. Функція пристосованості F є зворотною по відношенню до функції втрат S=E + αR, яка є мірою «непристосованості» моделі. Коефіцієнт α, який задається користувачем, змінює важливість і значимість кількості правил по відношенню до форми одержуваного рішення. При великих значеннях α рішення буде містити меншу кількість правил, забезпечуючи при цьому меншу точність моделі, а малі значення α приводять до протилежних результатів. На етапі постановки завдання можна задати максимально можливе значення кількості правил, і тоді в процесі пошуку моделі з великою кількістю правил будуть відкидатися.

Хромосома, яка містить фрагменти, що відповідають закодованому зображенню всіх параметрів та елементів структури нечіткої моделі, цікавих для користувача, називають *структурною хромосомою* або кодом структури моделі. На початку процесу пошуку довільним чином формують початкову популяцію структурних хромосом, яка, відповідно до процедури ГА, піддається еволюції. При цьому оцінка ступеня пристосованості хромосоми може являти собою окрему проблему, оскільки кожна хромосома задає лише структуру моделі, не визначаючи її оптимальних параметрів. Тому пошук оптимальних параметрів для кожної хромосоми в популяції стає окремою самостійною задачею, яку можна розглядати як локальну оптимізацію в межах структури, що задається конкретною хромосомою.

Література

1. Борисов А. Н., Крумберг О. А., Федоров И. П. Принятие решений на основе нечетких моделей: Примеры использования. – Рига: Зинатне, 1990. – 184с.

2. Борисов В. В., Круглов В. В., Федулов А. С. Нечеткие модели и сети. М.: Горячая линия - Телеком, 2007. - 284 с.

3. Добровська Л.М. Експертні системи в медицині [Текст]: метод. вказівки до практичних занять з дисципліни для студентів IV-го курсу між університетського медико-інженерного факультету / Уклад. Л.М. Добровська. - К.: НТУУ «КПІ», 2008. - 116 с.

4. Добровська Л.М. Експертні системи в медицині: навч. посіб. /Л.М. Добровська. - К.: Арстей, 2008. - 144 с.

5. Добровська Л. М. Теорія та практика нейронних мереж: навч. посіб. / Л.М. Добровська, І. А. Добровська. – К.: НТУУ «КПІ» Вид-во «Політехніка», 2015. – 396 с.

6. Леоненков А. Нечеткое моделирование в среде MATLAB и fuzzyTECH. – СПб.: БХВ-Петербург, 2005. – 736 с.

7. Пегат А. Нечеткое моделирование и управление / А. Пегат; пер.с англ. – М.: БИНОМ. Лаборатория знаний (Адаптивные и интеллектуальные системы), 2009.–798с.

8. Пономарев А.С. Нечеткие множества в задачах автоматизированного управления и принятия решений, 2005

http://ru.wikipedia.org/wiki/%D0%A5%D0%BE%D0%BB%D0%B5%D1%81%D1%82%D0%B5%D1%80%D0%B8%D0%BD

9. Потемкин В.Г. Вычисления в среде MATLAB. – М: Диалог-МИФИ, 2004. – 720с.

10. Ротштейн А. П. Медицинская диагностика на нечеткой логике. – Винница: Континент-ПРИМ. – 1996. – 132с.

11. Ротштейн О. П., Катєльніков Д.І. Ідентифікація нелінійних об’єктів нечіткими базами знань //Вісник ВШ. – 1997.– №4. – С. 98–103.

12. Ротштейн А. П., Штовба С. Д. Проектування нечітких баз знань. – Вінниця: Континент-ПРИМ, 1999. – 64 с.

13. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский. - Москва: Горячая Линия - Телеком, 2013. - 385 с.

14. Cordon Oscar, Herrera Francisco, Hoffmann Frank, Magdalena Luis

Genetic Fuzzy systems. Evolutionary tuning and learning of fuzzy knowledge bases. – World Scientific. – Singapore, New Jersey, London, Hong Kong. – 462 p.

15. Dobrovska L. Design of the universal classifier as a RBF network based on the CART solution tree // L. Dobrovska, I. Dobrovska / Eastern-European journal of enterprise technologies 4/4 (88) 2017. – 63-71 pp.

16. Fuzzy sets and fuzzy logic: Theory and Applications /George J. Klir and Bo Yuan. – USA, 1995. – 574 p.

17. Jang, Jyh–Shing Roger

Neuro-fuzzy and soft computing: a computation approach to learning and machine intelligence / Jyh-Shing Roger Jang, Chuen-Tsai Sun, Eiji Mizutani. – USA, 1997. – 640 p.

18. Steeb W.–H. The nonlinear workbook. – Third edition. – USA: World Scientific Publishing, 2005. – 588 p.

ДОДАТКИ

Додаток А

Основні функції пакету Fuzzy Logic Toolbox

(для розробки та подальшого використання систем нечіткого виведення)

Addmf додає нову функцію належності до системи нечіткого виведення FIS.

Синтаксис: a=addmf(a,‘varType’,varlndex,‘mfName’,‘mfType’, itifParanis)

Опис. Функція належності може бути додана тільки до існуючої змінної FIS. При цьому номери або індекси функціям належності призначаються в тому порядку, в якому вони додаються. Функція використовує такі шість аргументів:

a – ім’я змінної структури FIS;

‘varType’ – рядок, який зображує тип змінної, до якої додається функція належності. Може приймати одне із двох значень: ‘input’ або ‘output’;

varlndex – індекс змінної, до якої додається функція належності;

‘mfName’ – рядок, який зображує ім’я нової функції належності;

‘mfType’ – рядок, який зображує тип нової функції належності;

mf Params – вектор параметрів, які визначають функцію належності, яка додається.

Addrule додає нове правило до системи нечіткого виведення FIS.

Синтаксис: a=addrule (a, ruleList)

Опис. Функція addrule має два аргументи. Перший аргумент *a* – ім’я змінної FIS, другий аргумент ruleList – матриця, кожний рядок якої зображує деяке правило. Якщо система виведення має *m* вхідних і *n* вихідних змінних, то матриця ruleList повинна мати в точності (*m*+*n*+*2*) стовпчиків.

Перші *m* стовпчиків відносяться до вхідних змінних системи (при цьому номер стовпчика повинен відповідати індексу функції належності для конкретної вхідної змінної). Наступні *n* стовпчиків відносяться до вихідних змінних системи виведення (при цьому кожний стовпчик також має номер, який повинен відповідати індексу функції належності для вихідної змінної).

Стовпчик з номером (*m*+*n*+1) містить вагу, з якою застосовується дане правило. У загальному випадку доцільно задавати вагу, яка дорівнює «1».

Стовпчик з номером (*m*+*n*+*2*) містить число:

1, якщо для підвисновків даного правила використовується нечіткий оператор AND (нечітке І);

2, якщо для підумов даного правила використовується нечіткий оператор OR (нечітке АБО).

Addvar додає нову змінну до системи нечіткого виведення FIS.

Синтаксис: a = addvar(a, ‘varType’, ‘varName’, varBounds)

Опис. Функція використовує 4 аргументи:

a – ім’я структури FIS;

‘varType’ – рядок, який зображує тип змінної, яка додається: ‘input’ або ‘output’;

‘varName’ – рядок, який зображує ім’я змінної, яка додається;

varBounds – вектор, який задає границі області визначення змінної, що додається.

Номери або індекси змінним призначаються в тому порядку, в якому вони додаються. Нумерація вхідних і вихідних змінних виконується незалежно одна від одної.

Defuzz виконує дефаззифікацію функції належності.

Синтаксис: out = defuzz (x, mf, type)

Опис. Функція defuzz (x, mf, type) дозволяє одержати число, яке є результатом дефаззифікації функції належності mf для відповідної змінної *x*. При цьому може бути використаний один із методів дефаззифікації, який визначається аргументом type: ‘centroid’, ‘bisector’; ‘som’; ‘ lom’; ‘mom’.

Якщо для аргументу type не зазначено жодне з перерахованих вище значень, то передбачається використання метода дефаззифікації, який визначено самим користувачем. Це можна зробити шляхом додавання в m-

файл замість функції defuzz.m операторів, які реалізують додатковий метод дефаззифікації.

Evalfis виконує нечітке виведення в системі FIS.

Синтаксис: output= evalfis(input, fismat); output= evalfis (input, fismat, numPts); [output, IRR, ORR, ARR] = evalfis(input, fismat); [output, IRR, ORR, ARR] = evalfis(input, fismat, numPts)

Опис. Функція evalfis має такі аргументи:

input – число або матриця, яка визначає вхідні змінні. Якщо input є матрицею порядку ($m \times n$), де n – число вхідних змінних, то функція evalfis сприймає кожний рядок input як вектор, який відповідає окремому набору значень вхідних змінних. Функція повертає матрицю порядку ($m \times l$), в якій кожний рядок відповідає вектору значень вихідних змінних, де l – число всіх вихідних змінних системи;

fismat – ім'я структури FIS, для якої виконується нечітке виведення;

numPts – необов'язковий аргумент, який задає загальну кількість обраних точок, в яких оцінюються функції належності в областях визначення вхідних і вихідних змінних. Якщо цей аргумент не зазначений, то за замовчуванням використовується значення 101.

Функція evalfis повертає output – матрицю виходів розміром ($m \times l$).

Наступні вихідні значення функції є необов'язковими й повертаються функцією evalfis тільки в тому випадку, коли вказаний один набір значень входів ($m=1$):

IRR – результат обчислення значень функцій належності для відповідних вхідних змінних. Приймає вигляд матриці розміру ($numRules \times n$), де $numRules$ – загальна кількість правил в системі FIS;

ORR – результат обчислення значень ФН для відповідних вихідних змінних. Приймає вигляд матриці розміру ($numPts \times numRules * l$);

ARR – матриця розміру $(numPts \times l)$ агрегованих значень в області визначення кожної вихідної змінної.

Якщо функція викликається для одного набору значень вхідних змінних, то як результат повертається вектор значень вихідних змінних.

Evalmf задає одну з вбудованих функцій належності.

Синтаксис: y = evalmf (x, mfParams, mfType)

Опис. Функція evalmf дозволяє задати будь-яку з наявних в системі MatLab функцій належності, тип якої встановлюється рядком mfType. При цьому значення аргумента *x* встановлює область визначення ФН, яка задається, і повинна бути попередньо визначена. Рядок mfType повинен відповідати імені m-файла, в якому визначена ФН. Другий аргумент mfParams зображує собою вектор параметрів, які визначають ФН.

При необхідності користувач може задати власну ФН. Для цього її необхідно визначити в окремому m-файлі як функцію й використати ім'я цього файлу як третій аргумент функції evalmf. При цьому другий аргумент mfParams повинен відповідати параметрам цієї функції належності.

Fuzzy викликає редактор систем нечіткого виведення FIS.

Синтаксис: fuzzy; fuzzy(fismat)

Опис. Ця функція надає користувачеві можливість редагувати на високому рівні властивості СНВ, такі як кількість вхідних і вихідних змінних, метод дефаззифікації, який використовується тощо.

Призначення пунктів меню редактора FIS:

I. *Пункт меню File редактора FIS містить наступні операції:*

- New FIS... – дозволяє обрати тип нової системи нечіткого виведення, яка задається: Mamdani або Sugeno. При цьому СНВ, яка задається, не має ані вхідних, ані вихідних змінних, її ім'я задається за замовчуванням як Untitled;
- Import – дозволяє завантажувати в редактор FIS існуючу СНВ одним із таких способів: From Workspace... – з робочого простору програми MatLab або From Disk... – із зовнішнього файлу;
- Export – дозволяє зберегти відредаговану систему нечіткого

виведення одним із наступних способів: To Workspace... – у робочому просторі програми MatLab або To Disk... – у зовнішньому файлі;

- Print – дозволяє роздрукувати на принтері відредаговану систему нечіткого виведення;
- Close – закриває редактор FIS.

II. *Пункт меню Edit містить наступні операції:*

- Undo – відміняє виконання останньої дії;
- Add Variable... – дозволяє додати до системи нечіткого виведення, яка редагується, змінну одного з наступних типів: Input – вхідну або Output – вихідну;
- Remove Selected Variable – видаляє обрану змінну з системи нечіткого виведення, яка редагується;
- Membership Functions... – викликає редактор функцій належності;
- Rules – викликає програму редагування правил.

III. *Пункт меню View містить наступні операції:*

- rules – викликає програму перегляду правил;
- surface – викликає програму перегляду поверхні виведення.

У лівій нижній частині робочого інтерфейсу редактора FIS розташовані 5 меню, які розкриваються:

1. And method (Метод логічної кон'юнкції) – дозволяє задати один з наступних методів для виконання логічної кон'юнкції в умовах нечітких правил:

- min – метод мінімального значення;
- prod – метод алгебраїчного добутку;
- custom – метод, визначений самим користувачем.

2. Or method (Метод логічної диз'юнкції) – дозволяє задати один з наступних методів для виконання логічної диз'юнкції в умовах нечітких правил:

- max – метод максимального значення;

- probor – метод алгебраїчної суми;
- custom – метод, визначений самим користувачем.

3. Implication method (Метод виведення висновку) – дозволяє задати один із таких методів для виконання (активізації) логічного висновку в кожному із нечітких правил:

- min – метод мінімального значення;
- prod – метод алгебраїчного добутку;
- custom – метод, визначений користувачем.

4. Aggregation method (Метод агрегування) – дозволяє задати один із наступних методів для агрегування значень ФН кожної з вихідних змінних у висновках нечітких правил:

- max – метод максимального значення;
- sum – метод граничної суми;
- probor – метод алгебраїчної суми;
- custom – метод, визначений користувачем.

5. Denazification method (Метод дефаззифікації) – дозволяє задати один із таких методів для виконання дефаззифікації вихідних змінних в СНВ Мамдані:

- centroid – метод центру ваги для дискретної множини значень функції належності;
- bisector – метод центру площини (модифікація);
- mom (middle of maximum) – метод середнього максимуму, визначений як середнє арифметичне лівого і правого модальних значень;
- som (smallest of maximum) – метод лівого модального значення;
- lorn (largest of maximum) – метод правого модального значення;
- custom – метод, визначений самим користувачем.

Для СНВ Сугено можна обрати один з таких методів дефаззифікації:

- wtaver (weighted average) – метод зваженого середнього;
- wtsum (weighted sum) – метод зваженої суми.

***Gaussmf* –** внутрішня П-подібна ФН типу функції Гауса.

Синтаксис: y = gaussmf (x, [sig c])

Опис. Симетрична функція Гауса описує щільність нормального розподілу й визначається як: $\mu F(x) = e^{-\frac{(x-c)^2}{2\sigma^2}}$. Ця функція має два параметри: sig (або σ) і c, які задаються в формі вектора [sig c].

Gensurf генерує поверхню нечіткого виведення FIS.

Синтаксис: gensurf (fis); gensurf(fis, inputs, output); gensurf(fis, inputs, output, grids); gensurf(fis, inputs, output, grids, refinput)

Опис. Функція gensurf (fis) дозволяє одержати зображення поверхні нечіткого виведення для однієї з вихідних змінних структури СНВ FIS з ім'ям fis. В цьому форматі для побудови поверхні виведення використовуються перші дві вхідні змінні та перша вихідна змінна структури fis.

Функція gensurf (fis, inputs, output) у другому форматі дозволяє одержати зображення поверхні нечіткого виведення для однієї або двох вхідних змінних, номери яких задаються вектором inputs, і одної вихідної змінної з номером output для структури СНВ FIS з ім'ям fis.

Функція gensurf (fis, inputs, output, grids) в третьому форматі дозволяє одержати зображення сітки на рисунку поверхні нечіткого виведення для першої горизонтальної змінної й другої вертикальної вхідної змінної. При цьому номери змінних задаються вектором grids.

Функція gensurf (fis, inputs, output, grids, refinput) в четвертому форматі дозволяє одержати зображення поверхні нечіткого виведення для більш, ніж двох вихідних змінних. В цьому випадку вектор refinput задає номери вхідних змінних, які вважаються не такими, що змінюються.

Якщо функцію запустили в форматі [x, y, z]=gensurf(...), то в цьому випадку вона повертає значення змінних, які визначають поверхню виведення й можуть бути використані для зображення цієї поверхні засобами графіки системи MatLab.

Getfis дозволяє відобразити різні властивості СНВ FIS.

Синтаксис: getfis (a); getfis(a, ‘fisprop’); getfis(a,’vartype’,varindex, ‘varprop’); getfis (a,’vartype’, varirdex, ‘mf’, mfindex); getfis(a,’vartype’, varindex, ‘mf ‘, mfindex, ‘mfprop’)

Опис. Це основна функція, призначена для відображення у вікні команд окремих властивостей структури системи нечіткого виведення FIS. Функція getfis може бути використана в одному з п’яти зазначених форматів з такими аргументами:

a – ім’я структури FIS;

‘vartype’ – рядок із вказівкою типу змінної, що відображається (input або output);

varindex – ціле число, яке відповідає номеру змінної, що відображається;

‘mf’ – рядок, який відображає інформацію про функції належності;

mfindex – ціле число, яке відповідає номеру функції належності, що відображається.

Mfedit викликає графічний інтерфейс редактора ФН СНВ FIS.

Синтаксис: mfedit (‘a’)

Опис. Ця функція, записана у форматі mfedit(‘a’), викликає редактор функцій належності, який дозволяє користувачеві в графічному режимі аналізувати й модифікувати всі ФН деякої структури FIS – a.fis.

Призначення пунктів меню редактора функцій належності:

I. Пункт меню File редактора ФН містить такі ж команди, що й відповідний пункт меню редактора FIS.

II. Пункт меню Edit містить такі операції:

- Undo – скасовує виконання останньої дії;
- Add MF... – дозволяє додати вбудовану ФН термів для обраної змінної;
- Add Custom MF... – дозволяє додати ФН користувача для окремої змінної;

- Remove Current MF – дозволяє видалити окрему функцію належності;
- Remove All MFs – дозволяє видалити всі ФН для окремої змінної;
- FIS Properties... – викликає графічний інтерфейс редактора FIS;
- Rules... – викликає графічний інтерфейс редактора правил FIS.

Пункт меню View містить такі операції:

- Rules – викликає програму перегляду правил FIS;
- Surface – викликає програму перегляду поверхні виведення FIS.

Newfis створює нову систему нечіткого виведення FIS.

Синтаксис:

a=newfis(fisName, fisType, andMethod, orMethod,…

impMethod, aggMethod, defuzzMethod)

Опис. Ця функція призначена для створення нових структур СНВ FIS. Функція newfis має сім аргументів і повертає єдине значення – структуру FIS з ім’ям a. Аргументами функції є:

fisName – рядок з ім’ям структури FIS, при цьому також створюється файл з ім’ям fisName.fis;

fisType – тип системи нечіткого виведення FIS;

andMethod, orMethod, impMethod, aggMethod, defuzzMethod – відповідно задають методи для виконання нечітких логічних операцій AND, OR, імплікації, агрегування й дефаззифікації. Якщо ці аргументи не зазначені, то задаються відповідні їм значення за замовчуванням.

Plotfis *з*ображує діаграму системи нечіткого виведення FIS.

Синтаксис: plotfis (fismat)

Опис. Ця функція зображує діаграму верхнього рівня СНВ FIS з ім’ям fismat. Вхідні змінні й їхні функції належності розташовуються ліворуч від структурних характеристик FIS, в той час як вихідні змінні та їхні функції належності розташовуються праворуч.

Plotmf *з*ображує графіки всіх функцій належності для заданої змінної.

Синтаксис: plotmf (fismat, varType, varlndex)

Опис. Ця функція зображує графіки всіх ФН для деякої змінної із системи нечіткого виведення FIS з ім'ям fismat, яка є першим аргументом цієї функції. Другим і третім аргументами є тип і номер відповідної змінної: varType('input' або 'output') і varIndex. Ця функція також може бути використана разом із функцією subplot.

Readfis завантажує систему нечіткого виведення FIS із зовнішнього файлу.

Синтаксис: fismat = readfis ('filename')

Опис. Ця функція зчитує СНВ із зовнішнього файлу з ім'ям filename, яке має розширенння fis, і завантажує його дані в робочу область. Якщо дана функція викликається без аргументу, то відкривається стандартне діалогове вікно відкриття файлу.

Ruleedit викликає графічний інтерфейс редактора правил СНВ.

Синтаксис: ruleedit ('a'); ruleedit(a)

Опис. Ця функція, записана у форматі ruleedit ('a'), викликає редактор правил, який дозволяє користувачеві в графічному режимі аналізувати й модифікувати правила продукцій системи нечіткого виведення FIS, що зберігається у зовнішньому файлі з ім'ям a.fis. Вона дозволяє також виконувати граматичний аналіз правил, які використовуються в деякій системі нечіткого виведення FIS.

Призначення пунктів меню редактора правил системи нечіткого виведення:

I. Пункт меню File редактора правил містить такі ж операції, що й відповідний пункт меню редактора FIS.

II. Пункт меню Edit містить такі операції:

- Undo – скасовує виконання останньої дії;
- FIS Properties... – викликає редактор FIS;
- Membership Functions... – викликає редактор функцій належності.

III. Пункт меню View містить такі операції:

- Rules – викликає програму перегляду правил;
- Surface – викликає програму перегляду поверхні виведення.

IV. Пункт меню Options містить такі команди:

- Language – дозволяє обрати мову для запису правил у формі тексту (English, Deutsch або Francais);
- Format – дозволяє обрати формат запису правил CHB: Verbose(у формі тексту), Symbolic (у символьній формі) або Indexed (у цифровій формі).

Ruleview – викликає графічний інтерфейс програми перегляду правил системи нечіткого виведення FIS.

Синтаксис: ruleview (‘a’); ruleview(a)

Опис. Ця функція, записана у форматі ruleview(‘a’), викликає програму перегляду правил, яка зображує діаграму нечіткого виведення для структури FIS, що зберігається у зовнішньому файлі з ім’ям a.fis.

Призначення пунктів меню програми перегляду правил:

I. Пункт меню File редактора правил містить такі ж операції, що й відповідний пункт меню редактора FIS.

II. Пункт меню Edit містить такі операції:

- Undo – скасовує виконання останньої дії;
- FIS Properties... – викликає редактор FIS;
- Membership Functions... – викликає редактор ФН;
- Rules – викликає програму редагування правил.

III. Пункт меню View містить такі операції:

Surface – викликає програму перегляду поверхні виведення.

IV. Пункт меню Options містить такі операції:

Format – дозволяє обрати формат запису правил CHB: Verbose (у формі тексту), Symbolic (у символьній формі) або Indexed (у цифровій формі).

Якщо клацнути на номері правила в лівій частині діаграми нечіткого виведення, то відповідне правило з’явиться в рядку стану в нижній частині

графічного інтерфейсу програми перегляду правил в тому форматі, який був обраний операцією Options→Format.

Приклад використання функції ruleview: ruleview (‘tipper’)

Див. також: fuzzy, mfedit, ruleed.it, surfview.

Setfis задає властивості системи нечіткого виведення FIS.

Синтаксис: a = setfis (a, ‘fispropname’, ‘newfisprop’);

a = setfis(a, ‘vartype’, varindex, ‘varpropname’, ‘newvarprop’);

a=setfis(a, ‘vartype’, varindex, ‘mf’, mfindex, ‘mfpropname’, ‘newmfprop’)

Опис. Функція setfis може бути викликана із трьома, п’ятьма або сімома аргументами в залежності від того, чи необхідно задати структуру FIS в цілому, або тільки змінну структури FIS, або окрему ФН для однієї із змінних структури FIS. Аргументи функції приймають такі значення:

a – ім’я структури FIS;

‘vartype’ – рядок із вказуванням типу змінної (input або output);

varindex – номер вхідної або вихідної змінної;

‘mf’ – рядок, необхідний для виклику функції із п’ятьма або сімома аргументами, із вказівкою на те, що функція setfis задає ФН для деякої змінної;

mfindex – номер функції належності для обраної змінної;

‘fispropname’ – рядок із вказуванням властивості поля структури FIS, яке може приймати такі значення: name, type, andmethod, ormethod, impmethod, aggmethod, defuzzmethod;

‘newfisprop’ – рядок із вказуванням імені властивості FIS або метода, який задається данною функцією;

‘varpropname’ – рядок із вказуванням імені поля структури FIS, яке задається даною функцією (name або range);

‘newvarprop’ – рядок із вказуванням імені змінної, яка задається, (для name) або діапазону зміни її значень (для range);

‘mfpropname’ – рядок із вказуванням імені поля функції належності, яка задається (name, type або params);

‘newmfprop’ – рядок із вказуванням імені або типу поля ФН, яка задається (для name або type) або вектор із вказуванням параметрів ФН, яка задається (params).

Showfis відображає структуру системи нечіткого виведення FIS.

Синтаксис: showfis (fismat)

Опис. Функція showfis (fismat) виводить у вікні команд системи MatLab значення всіх полів структури fismat, яка визначає СНВ FIS і повинна знаходитися в робочій області.

Showrule відображає правила системи нечіткого виведення FIS.

Синтаксис: showrule (fis); showrule (fis, indexList); showrule (fis, indexList, format); showrule (fis, indexList, format, Lang)

Опис. Ця функція використовується для візуалізації правил СНВ FIS. Вона може бути викликана з різним числом аргументів (від 1 до 4):

fis – обов’язковий аргумент, який повинен відповідати імені структури СНВ FIS;

indexList – необов’язковий аргумент, який зображується у вигляді вектора номерів правил, що потрібно відобразити у вікні команд;

format – необов’язковий аргумент, який зображується у вигляді рядка формату, у якому потрібно відобразити правила. Цей рядок може приймати одне із таких значень: ‘verbose’ (значення за замовчуванням, яке відповідає відображенню правил у формі тексту англійскої мови), ‘symbolic’ (відповідає відображенню правил в символьній формі), ‘indexed’ (відповідає відображенню правил в цифровій формі, при цьому вказуються тільки номери правил);

Lang – необов’язковий аргумент, який зображується у вигляді рядка із вказуванням мови відображення правил у вікні команд у випадку використання формату ‘verbose’. Цей рядок може приймати одне із наступних значень: ‘english’, ‘francais’ або ‘deutsch’.

Surfview викликає графічний інтерфейс програми перегляду поверхні СНВ.

Синтаксис: surfview (‘a’); surfview(a)

Опис. Ця функція, записана в форматі surfview(‘a’), викликає програму перегляду поверхні, яка зображує поверхню нечіткого виведення для структури FIS, збереженної у зовнішньому файлі з ім’ям a.fis, для будь-якої однієї або двох із її вхідних змінних.

Призначення пунктів меню програми перегляду поверхні виведення:

I. Пункт меню File редактора правил містить такі ж операції, що й відповідний пункт меню редактора FIS.

II. Пункт меню Edit містить наступні операції:

- Undo – скасовує виконання останньої дії;
- FIS Properties... – викликає редактор FIS;
- Membership Functions... – викликає редактор функцій належності;
- Rules – викликає програму редагування правил.

III.Пункт меню View містить наступні операції:

- Rules – викликає програму перегляду правил.

IV.Пункт меню Options містить наступні операції:

- Plot – дозволяє обрати один із 8 стилів зображення графіка поверхні виведення;
- Color Map – дозволяє обрати одну із 4 схем кольорів зображення графіка поверхні виведення;
- Always evaluate – активізація цього пункта меню призводить до автоматичного формування нової поверхні виведення щораз, коли вносяться такі зміни в систему нечіткого виведення, які впливають на форму графіка поверхні виведення (наприклад, зміни кількості точок сітки графіка). Це значення прийнято за замовчуванням.

Trimf – вбудована трикутна функція належності.

*Синтаксис:*y = trimf(x, [a b c])

Опис. Ця функція використовується для задання трикутних ФН. Функція trimf використовує три аргументи a, b і c, які приймають довільні дійсні значення й упорядковані відношенням: $a \le b \le c$, де b – модальне значення нечіткої множини.

Writefis зберігає СНВ FIS у зовнішньому файлі.

Синтаксис: writefis (fismat); writefis (fismat, 'filename');

writefis (fismat, 'filename', 'dialog')

Опис. Функція writefis зберігає структуру СНВ FIS з ім'ям fismat, яка знаходиться в робочій області системи MatLab, у зовнішньомі файлі з розширенням fis. Виклик цієї функції в форматі writefis(fismat) відкриває стандартне діалогове вікно збереження файлу. Використання функції в форматі writefis(fismat, 'filename') записує структуру системи нечіткого виведення FIS з ім'ям fismat, яка знаходиться в робочій області системи MatLab, у зовнішньому файлі з розширенням fis і ім'ям filename. При цьому файл зберігається в поточному каталозі.

Виклик функції в форматі writefis(fismat, 'filename', 'dialog') відкриває діалогове вікно збереження файлу, при цьому запис буде виконуватися у файл з ім'ям filename.fis.

Додаток Б

Синтаксис деяких операторів MatLab

Оператори керування процесом виконання операторів m-файлу використовують головним чином для організації обчислювального процесу, який записується у вигляді деякої програми (m-файлу). В системі MatLab реалізовано такі оператори управління процесом виконання операторів:

1. Оператор *if*

Синтаксис: if умоваінструкції end

if умоваінструкції1 else інструкції2 end

if умова1 інструкції1 elseif умова2 інструкції2 else інструкції3 end

Оператор умовного переходу обчислює умову в формі деякого логічного виразу і виконує відповідну групу інструкцій в залежності від значення цієї умови. Наведемо приклад оператора умовного переходу, який виконується тільки при парних значеннях змінної a:

```
if rem(a,2)==0 disp('a - парне') b=a/2; end
```

Приклад формування трьохдіагональної матриці:

```
n=4;
for i=1 : n
 for j=1 : n
  if i==j A(i,j)=2;elseif abs(i-j)==1 A(i,j)=1;else A(i,j)=0;
  end;
end; end; A
```

```
A =

     2     1     0     0
     1     2     1     0
     0     1     2     1
     0     0     1     2
```

2. Оператор *switch*

Синтаксис: switch <вираз>

% вираз – це скаляр або рядок

case <значення_1> інструкції

% виконуються, якщо <вираз> = <значення_1>

case {<значення_2>, <значення_3>, . . . , <значення_k>} інструкції

% виконуються, якщо <вираз> = <значення_1 (i = 2:k)>

otherwise інструкції

% виконуються, якщо <вираз> не співпав із жодним із значень end

Оператор *switch* виконує розгалуження в залежності від значень деякої змінної або виразу. Наприклад, наступний оператор *switch* перевіряє значення змінної *method*, і якщо це значення дорівнює ‘Bilinear’, то оператор виводить на екран повідомлення ‘Method is linear’:

```
method = 'Bilinear';
switch lower(method)
  case {'linear','bilinear'}
                disp('Method is linear')
  case 'cubic'
                disp('Method is cubic')
  otherwise
                disp('Unknown method.')
end
```

3. Оператор *for*

Синтаксис: for <індекс> = <ПЗ>: <крок>: <КЗ> <інструкції> end

Оператор циклу з визначеною кількістю ітерацій виконує інструкцію або групу інструкцій наперед визначену кількість разів. Тут <індекс> – індекс циклу; <ПЗ> і <КЗ> – початкове й кінцеве значення індекса: <крок> – приріст індекса циклу (за замовчуванням дорівнює 1). Наведемо приклад обчислення рангу магічної матриці в залежності від її порядку:

```
for i=3 : 12 r(i) = rank(magic(i)); end;
```

4. Оператор *while*

Синтаксис: while <логічний вираз> <інструкції> end

Оператор циклу з невизначеним числом ітерацій (з передумовою) *while* виконує інструкцію або групу інструкцій, поки логічний вираз приймає значення «*істина*».

Приклад: знайти число n, факторіал якого містить 100 знаків.

```
n = 1;
```

```
while prod(1:n)<1.e100
 n = n+1;
end
n
n = 70 % ВІДПОВІДЬ
```

5. Оператор *try.* Оператор «TRY оператори_1 CATCH оператори_2 END» змінює послідовність виконання операторів m-файлу, якщо виникає помилкова ситуація. Якщо в результаті спроби виконати групу операторів оператори_1 виникає особлива ситуація (збій), управління передається групі оператори_2 (обробка ситуацій збою). Якщо збій не виникає, група оператори_2 не виконується.

6. Оператор *break.* Функція break призупиняє виконання циклів *for* і *while*. Наприклад, наведений нижче оператор циклу *while* буде виконуватися до тих пір, поки не буде введено нульове або від'ємне значення змінної *n*.

```
while 1
 n = input('Введіть n. Зупинка циклу при n<=0, n = ')
 if n <= 0, disp('Робота завершена'), break, end
 r = rank(magic(n))
end

Введіть n. Зупинка циклу при n<=0, n = -1
n =  -1
Робота завершена.
```

7. Оператор *continue.* Функція *continue* припиняє виконання поточного кроку в циклах *for* і *while* й переходить на наступний крок. У випадку, коли цикли входять до складу один одного, здійснюється повернення до циклу більш високого рівня.

8. Оператор *return.* Команда return (достроковий вихід з тіла функції) виконує повернення до функції, яка її викликала, або до режиму роботи з клавіатурою, вона дозволяє також завершити режим роботи із клавіатурою.

Система MatLab орієнтована на включення користувача в процес розв'язання задачі. Організація інтерактивної взаємодії користувача з комп'ютером основана на таких базових операціях, як:

– виведення інформації та даних на екран терміналу;

– призупинення обчислень для аналізу числових і графічних даних;

– запит інформації від користувача про продовження або зупинку процесу обчислень або введення додаткових даних.

Розглянемо найпростіші оператори організації введення/виведення інформації в системі MatLab.

I. *Введення інформації користувачем*

Найпростіший спосіб введення числової і символьної інформації (яка набирається на клавіатурі) – використання функції *input*. Синтаксис цієї функції має вигляд:

x = input(‘<Запит>’) або x = input (‘<Запит>’ , ‘s’)

Функція x=input(‘<Запит>’) виводить на екран рядок із запитом і чекає введення виразу, допустимого в системі MatLab (арифметичного виразу, імені вбудованної функції або m-файлу), величина якого буде підрахована з урахуванням поточного стану змінних робочого простору і повернена як значення функції. Якщо функція має декілька вихідних параметрів, то вихідній змінній x присвоюється тільки перше значення. Якщо користувач у відповідь на запит натиснув тільки клавішу <Enter>, то функція повертає пустий масив даних типу double розміром 0×0. У другому випадку текст, який набирає користувач, сприймається як рядок символів, який і є значенням функції. В обох наведених випадках текст запит може включати в себе символи “*\n*”, які керують процесом перенесення курсора на початок наступного рядка, наприклад:

```
>>i = input('Продовжити обчислення? Так/Ні [Y]: ' , 's' )
Продовжити обчислення? Так/Ні [Y]:
Y
i = Y
```

II. Виведення результатів обчислень

1. Оператор *disp* (виведення значень змінних і тексту на екран)

Для виведення значень виразу достатньо не набирати після нього символ ";", в інших випадках можна скористатися функцією disp (від англ. display – відобразити).

Синтаксис: disp (<змінна>) або disp('<текст>')

Оператор disp(X) виводить на термінал значення змінної X без вказування імені. Оператор disp ('<текст>') виводить на термінал символьний рядок '<текст>'. Після кожної команди disp відбувається перехід курсору на новий рядок. Наприклад:

```
>>disp('Стовпчик_1 Стовпчик_2 Стовпчик_3');
disp (rand (5, 3))
```

```
Стовпчик_1   Стовпчик_2 Стовпчик_3
    0.4057    0.0579    0.2028
    0.9355    0.3529    0.1987
    0.9169    0.8132    0.6038
    0.4103    0.0099    0.2722
    0.8936    0.1389    0.1988
```

2. Оператори *fprintf, sprintf* (форматування даних для виведення на екран або запису в файл).

Синтаксис: count =fprintf (<дескриптор_файлу>,<формат>,A,...)

fprintf (<формат>, A,...)

s = sprintf (<формат>, A,...)

[s,<повідомлення_про_помилку>]=sprintf(<формат>, A, ...)

Функція count = fprintf (<дескриптор_файлу>, <формат>, A, ...) перетворює дані в рядки символів відповідно до вказаного формату і виводить їх на екран або в файл (в залежності від значення дескриптора файлу). Дескриптор файлу – це ціле число, яке приймає значення 1 для виведення на екран і 2 – у випадку стандартної помилки (за замовчуванням дескриптор файла дорівнює 1).

Команда fprintf (<формат>, A, ...) виводить дані на екран.

Функція s=sprintf (<формат>, A, ...) перетворює дані в рядки символів відповідно до вказаного формату і повертає їх у вигляді змінної s, але не записує їх у файл.

Функція: [s,<повідомлення_про_помилку>]=sprintf(<формат>, A, ...) повертає, окрім того, повідомлення про помилку, якщо остання мала місце, або порожній рядок.

Рядок <формат> визначає способи виведення на екран, кількість значущих цифр, ширину поля виведення й інші характеристики опису формату даних. Наприклад, оператор

```
>>fprintf('Довжина одиничного кола дорівнює %g \n', 2*pi)
```

виводить на екран рядок:

Довжина одиничного кола дорівнює6.28319.

Наведемо приклади використання оператора sprintf:

Команда	*Результат виконання*
sprintf (‘%0.5g’,(1+sqrt(5))/2)	1.618
sprintf (‘%15.5f ‘, 1/eps)	4503599627370496.00000
sprintf (‘%d ‘, round(pi))	3
sprintf (‘%s’, ‘hello’)	Hello
sprintf (‘Розмір масиву %d x %d.’, 2, 3)	Розмір масиву 2 x 3

3. Функції *error* або *warning* використовують для виведення повідомлення про помилку (функція error) або попередження (функція warning).

Додаток В

База даних діабет

Дані про населення, яке мешкає поряд із Фенікс (штат Арізона, США). Діагностика діабету відповідає критеріям Всесвітньої організації охорони здоров'я:

http://archive.ics.uci.edu/ml/datasets.html,

Кількість екземплярів: 768. Використані такі 8 атрибутів:

1. Кількість разів була вагітною.
2. Концентрація глюкози в плазмі через 2 год. після навантаження – тест на толерантність до глюкози
3. Діастолічний артеріальний тиск (мм рт.ст.)
4. Трицепс (товщина кожних складок): мм
5. 2-години інсуліну в сиворотці крові
6. Індекс маси тіла (вага в кг / (рост в м) ^ 2)
7. Функція діабет за родословною
8. Вік: років
9. Бінарна змінна *клас*: {0,1} (клас 1 означає "у хворого є діабет").

Клас Кількість екземплярів

0 500

1 268

Короткий статистичний аналіз:

Номер атрибута Середнє значення Стандартне відхилення:

1	3.8	3.4
2	120.9	32.0
3	69.1	19.4
4	20.5	16.0
5	79.8	115.2
6	32.0	7.9
7	0.5	0.3
8	33.2	11.8

Дані

6,148,72,35,0,33.6,0.627,50,1	3,180,64,25,70,34.0,0.271,26,0
1,85,66,29,0,26.6,0.351,31,0	7,133,84,0,0,40.2,0.696,37,0
8,183,64,0,0,23.3,0.672,32,1	7,106,92,18,0,22.7,0.235,48,0
1,89,66,23,94,28.1,0.167,21,0	9,171,110,24,240,45.4,0.721,54,1
0,137,40,35,168,43.1,2.288,33,1	7,159,64,0,0,27.4,0.294,40,0
5,116,74,0,0,25.6,0.201,30,0	0,180,66,39,0,42.0,1.893,25,1
3,78,50,32,88,31.0,0.248,26,1	1,146,56,0,0,29.7,0.564,29,0
10,115,0,0,0,35.3,0.134,29,0	2,71,70,27,0,28.0,0.586,22,0
2,197,70,45,543,30.5,0.158,53,1	7,103,66,32,0,39.1,0.344,31,1
8,125,96,0,0,0.0,0.232,54,1	7,105,0,0,0,0.0,0.305,24,0
4,110,92,0,0,37.6,0.191,30,0	1,103,80,11,82,19.4,0.491,22,0
10,168,74,0,0,38.0,0.537,34,1	1,101,50,15,36,24.2,0.526,26,0
10,139,80,0,0,27.1,1.441,57,0	5,88,66,21,23,24.4,0.342,30,0
1,189,60,23,846,30.1,0.398,59,1	8,176,90,34,300,33.7,0.467,58,1
5,166,72,19,175,25.8,0.587,51,1	7,150,66,42,342,34.7,0.718,42,0
7,100,0,0,0,30.0,0.484,32,1	1,73,50,10,0,23.0,0.248,21,0
0,118,84,47,230,45.8,0.551,31,1	7,187,68,39,304,37.7,0.254,41,1
7,107,74,0,0,29.6,0.254,31,1	0,100,88,60,110,46.8,0.962,31,0
1,103,30,38,83,43.3,0.183,33,0	0,146,82,0,0,40.5,1.781,44,0
1,115,70,30,96,34.6,0.529,32,1	0,105,64,41,142,41.5,0.173,22,0
3,126,88,41,235,39.3,0.704,27,0	2,84,0,0,0,0.0,0.304,21,0
8,99,84,0,0,35.4,0.388,50,0	8,133,72,0,0,32.9,0.270,39,1
7,196,90,0,0,39.8,0.451,41,1	5,44,62,0,0,25.0,0.587,36,0
9,119,80,35,0,29.0,0.263,29,1	2,141,58,34,128,25.4,0.699,24,0
11,143,94,33,146,36.6,0.254,51,1	7,114,66,0,0,32.8,0.258,42,1
10,125,70,26,115,31.1,0.205,41,1	5,99,74,27,0,29.0,0.203,32,0
7,147,76,0,0,39.4,0.257,43,1	0,109,88,30,0,32.5,0.855,38,1
1,97,66,15,140,23.2,0.487,22,0	2,109,92,0,0,42.7,0.845,54,0
13,145,82,19,110,22.2,0.245,57,0	1,95,66,13,38,19.6,0.334,25,0
5,117,92,0,0,34.1,0.337,38,0	4,146,85,27,100,28.9,0.189,27,0
5,109,75,26,0,36.0,0.546,60,0	2,100,66,20,90,32.9,0.867,28,1
3,158,76,36,245,31.6,0.851,28,1	5,139,64,35,140,28.6,0.411,26,0
3,88,58,11,54,24.8,0.267,22,0	13,126,90,0,0,43.4,0.583,42,1
6,92,92,0,0,19.9,0.188,28,0	4,129,86,20,270,35.1,0.231,23,0
10,122,78,31,0,27.6,0.512,45,0	1,79,75,30,0,32.0,0.396,22,0
4,103,60,33,192,24.0,0.966,33,0	1,0,48,20,0,24.7,0.140,22,0
11,138,76,0,0,33.2,0.420,35,0	7,62,78,0,0,32.6,0.391,41,0
9,102,76,37,0,32.9,0.665,46,1	5,95,72,33,0,37.7,0.370,27,0
2,90,68,42,0,38.2,0.503,27,1	0,131,0,0,0,43.2,0.270,26,1
4,111,72,47,207,37.1,1.390,56,1	2,112,66,22,0,25.0,0.307,24,0

3,113,44,13,0,22.4,0.140,22,0	0,162,76,56,100,53.2,0.759,25,1
2,74,0,0,0,0.0,0.102,22,0	6,111,64,39,0,34.2,0.260,24,0
7,83,78,26,71,29.3,0.767,36,0	2,107,74,30,100,33.6,0.404,23,0
0,101,65,28,0,24.6,0.237,22,0	5,132,80,0,0,26.8,0.186,69,0
5,137,108,0,0,48.8,0.227,37,1	0,113,76,0,0,33.3,0.278,23,1
2,110,74,29,125,32.4,0.698,27,0	1,88,30,42,99,55.0,0.496,26,1
13,106,72,54,0,36.6,0.178,45,0	3,120,70,30,135,42.9,0.452,30,0
2,100,68,25,71,38.5,0.324,26,0	1,118,58,36,94,33.3,0.261,23,0
15,136,70,32,110,37.1,0.153,43,1	1,117,88,24,145,34.5,0.403,40,1
1,107,68,19,0,26.5,0.165,24,0	0,105,84,0,0,27.9,0.741,62,1
1,80,55,0,0,19.1,0.258,21,0	4,173,70,14,168,29.7,0.361,33,1
4,123,80,15,176,32.0,0.443,34,0	9,122,56,0,0,33.3,1.114,33,1
7,81,78,40,48,46.7,0.261,42,0	3,170,64,37,225,34.5,0.356,30,1
4,134,72,0,0,23.8,0.277,60,1	8,84,74,31,0,38.3,0.457,39,0
2,142,82,18,64,24.7,0.761,21,0	2,96,68,13,49,21.1,0.647,26,0
6,144,72,27,228,33.9,0.255,40,0	2,125,60,20,140,33.8,0.088,31,0
2,92,62,28,0,31.6,0.130,24,0	0,100,70,26,50,30.8,0.597,21,0
1,71,48,18,76,20.4,0.323,22,0	0,93,60,25,92,28.7,0.532,22,0
6,93,50,30,64,28.7,0.356,23,0	0,129,80,0,0,31.2,0.703,29,0
1,122,90,51,220,49.7,0.325,31,1	5,105,72,29,325,36.9,0.159,28,0
1,163,72,0,0,39.0,1.222,33,1	3,128,78,0,0,21.1,0.268,55,0
1,151,60,0,0,26.1,0.179,22,0	5,106,82,30,0,39.5,0.286,38,0
0,125,96,0,0,22.5,0.262,21,0	2,108,52,26,63,32.5,0.318,22,0
1,81,72,18,40,26.6,0.283,24,0	10,108,66,0,0,32.4,0.272,42,1
2,85,65,0,0,39.6,0.930,27,0	4,154,62,31,284,32.8,0.237,23,0
1,126,56,29,152,28.7,0.801,21,0	0,102,75,23,0,0.0,0.572,21,0
1,96,122,0,0,22.4,0.207,27,0	9,57,80,37,0,32.8,0.096,41,0
4,144,58,28,140,29.5,0.287,37,0	2,106,64,35,119,30.5,1.400,34,0
3,83,58,31,18,34.3,0.336,25,0	5,147,78,0,0,33.7,0.218,65,0
0,95,85,25,36,37.4,0.247,24,1	2,90,70,17,0,27.3,0.085,22,0
3,171,72,33,135,33.3,0.199,24,1	1,136,74,50,204,37.4,0.399,24,0
8,155,62,26,495,34.0,0.543,46,1	4,114,65,0,0,21.9,0.432,37,0
1,89,76,34,37,31.2,0.192,23,0	9,156,86,28,155,34.3,1.189,42,1
4,76,62,0,0,34.0,0.391,25,0	1,153,82,42,485,40.6,0.687,23,0
7,160,54,32,175,30.5,0.588,39,1	8,188,78,0,0,47.9,0.137,43,1
4,146,92,0,0,31.2,0.539,61,1	7,152,88,44,0,50.0,0.337,36,1
5,124,74,0,0,34.0,0.220,38,1	2,99,52,15,94,24.6,0.637,21,0
5,78,48,0,0,33.7,0.654,25,0	1,109,56,21,135,25.2,0.833,23,0
4,97,60,23,0,28.2,0.443,22,0	2,88,74,19,53,29.0,0.229,22,0
4,99,76,15,51,23.2,0.223,21,0	17,163,72,41,114,40.9,0.817,47,1

4,151,90,38,0,29.7,0.294,36,0	0,113,80,16,0,31.0,0.874,21,0
7,102,74,40,105,37.2,0.204,45,0	1,138,82,0,0,40.1,0.236,28,0
0,114,80,34,285,44.2,0.167,27,0	0,108,68,20,0,27.3,0.787,32,0
2,100,64,23,0,29.7,0.368,21,0	2,99,70,16,44,20.4,0.235,27,0
0,131,88,0,0,31.6,0.743,32,1	6,103,72,32,190,37.7,0.324,55,0
6,104,74,18,156,29.9,0.722,41,1	5,111,72,28,0,23.9,0.407,27,0
3,148,66,25,0,32.5,0.256,22,0	8,196,76,29,280,37.5,0.605,57,1
4,120,68,0,0,29.6,0.709,34,0	5,162,104,0,0,37.7,0.151,52,1
4,110,66,0,0,31.9,0.471,29,0	1,96,64,27,87,33.2,0.289,21,0
3,111,90,12,78,28.4,0.495,29,0	7,184,84,33,0,35.5,0.355,41,1
6,102,82,0,0,30.8,0.180,36,1	2,81,60,22,0,27.7,0.290,25,0
6,134,70,23,130,35.4,0.542,29,1	0,147,85,54,0,42.8,0.375,24,0
2,87,0,23,0,28.9,0.773,25,0	7,179,95,31,0,34.2,0.164,60,0
1,79,60,42,48,43.5,0.678,23,0	0,140,65,26,130,42.6,0.431,24,1
2,75,64,24,55,29.7,0.370,33,0	9,112,82,32,175,34.2,0.260,36,1
8,179,72,42,130,32.7,0.719,36,1	12,151,70,40,271,41.8,0.742,38,1
6,85,78,0,0,31.2,0.382,42,0	5,109,62,41,129,35.8,0.514,25,1
0,129,110,46,130,67.1,0.319,26,1	6,125,68,30,120,30.0,0.464,32,0
5,143,78,0,0,45.0,0.190,47,0	5,85,74,22,0,29.0,1.224,32,1
5,130,82,0,0,39.1,0.956,37,1	5,112,66,0,0,37.8,0.261,41,1
6,87,80,0,0,23.2,0.084,32,0	0,177,60,29,478,34.6,1.072,21,1
0,119,64,18,92,34.9,0.725,23,0	2,158,90,0,0,31.6,0.805,66,1
1,0,74,20,23,27.7,0.299,21,0	7,119,0,0,0,25.2,0.209,37,0
5,73,60,0,0,26.8,0.268,27,0	7,142,60,33,190,28.8,0.687,61,0
4,141,74,0,0,27.6,0.244,40,0	1,100,66,15,56,23.6,0.666,26,0
7,194,68,28,0,35.9,0.745,41,1	1,87,78,27,32,34.6,0.101,22,0
8,181,68,36,495,30.1,0.615,60,1	0,101,76,0,0,35.7,0.198,26,0
1,128,98,41,58,32.0,1.321,33,1	3,162,52,38,0,37.2,0.652,24,1
8,109,76,39,114,27.9,0.640,31,1	4,197,70,39,744,36.7,2.329,31,0
5,139,80,35,160,31.6,0.361,25,1	0,117,80,31,53,45.2,0.089,24,0
3,111,62,0,0,22.6,0.142,21,0	4,142,86,0,0,44.0,0.645,22,1
9,123,70,44,94,33.1,0.374,40,0	6,134,80,37,370,46.2,0.238,46,1
7,159,66,0,0,30.4,0.383,36,1	1,79,80,25,37,25.4,0.583,22,0
11,135,0,0,0,52.3,0.578,40,1	4,122,68,0,0,35.0,0.394,29,0
8,85,55,20,0,24.4,0.136,42,0	3,74,68,28,45,29.7,0.293,23,0
5,158,84,41,210,39.4,0.395,29,1	4,171,72,0,0,43.6,0.479,26,1
1,105,58,0,0,24.3,0.187,21,0	7,181,84,21,192,35.9,0.586,51,1
3,107,62,13,48,22.9,0.678,23,1	0,179,90,27,0,44.1,0.686,23,1
4,109,64,44,99,34.8,0.905,26,1	9,164,84,21,0,30.8,0.831,32,1
4,148,60,27,318,30.9,0.150,29,1	0,104,76,0,0,18.4,0.582,27,0

1,91,64,24,0,29.2,0.192,21,0	0,146,70,0,0,37.9,0.334,28,1
4,91,70,32,88,33.1,0.446,22,0	10,129,76,28,122,35.9,0.280,39,0
3,139,54,0,0,25.6,0.402,22,1	7,133,88,15,155,32.4,0.262,37,0
6,119,50,22,176,27.1,1.318,33,1	7,161,86,0,0,30.4,0.165,47,1
2,146,76,35,194,38.2,0.329,29,0	2,108,80,0,0,27.0,0.259,52,1
9,184,85,15,0,30.0,1.213,49,1	7,136,74,26,135,26.0,0.647,51,0
10,122,68,0,0,31.2,0.258,41,0	5,155,84,44,545,38.7,0.619,34,0
0,165,90,33,680,52.3,0.427,23,0	1,119,86,39,220,45.6,0.808,29,1
9,124,70,33,402,35.4,0.282,34,0	4,96,56,17,49,20.8,0.340,26,0
1,111,86,19,0,30.1,0.143,23,0	5,108,72,43,75,36.1,0.263,33,0
9,106,52,0,0,31.2,0.380,42,0	0,78,88,29,40,36.9,0.434,21,0
2,129,84,0,0,28.0,0.284,27,0	0,107,62,30,74,36.6,0.757,25,1
2,90,80,14,55,24.4,0.249,24,0	2,128,78,37,182,43.3,1.224,31,1
0,86,68,32,0,35.8,0.238,25,0	1,128,48,45,194,40.5,0.613,24,1
12,92,62,7,258,27.6,0.926,44,1	0,161,50,0,0,21.9,0.254,65,0
1,113,64,35,0,33.6,0.543,21,1	6,151,62,31,120,35.5,0.692,28,0
3,111,56,39,0,30.1,0.557,30,0	2,146,70,38,360,28.0,0.337,29,1
2,114,68,22,0,28.7,0.092,25,0	0,126,84,29,215,30.7,0.520,24,0
1,193,50,16,375,25.9,0.655,24,0	14,100,78,25,184,36.6,0.412,46,1
11,155,76,28,150,33.3,1.353,51,1	8,112,72,0,0,23.6,0.840,58,0
3,191,68,15,130,30.9,0.299,34,0	0,167,0,0,0,32.3,0.839,30,1
3,141,0,0,0,30.0,0.761,27,1	2,144,58,33,135,31.6,0.422,25,1
4,95,70,32,0,32.1,0.612,24,0	5,77,82,41,42,35.8,0.156,35,0
3,142,80,15,0,32.4,0.200,63,0	5,115,98,0,0,52.9,0.209,28,1
4,123,62,0,0,32.0,0.226,35,1	3,150,76,0,0,21.0,0.207,37,0
5,96,74,18,67,33.6,0.997,43,0	2,120,76,37,105,39.7,0.215,29,0
0,138,0,0,0,36.3,0.933,25,1	10,161,68,23,132,25.5,0.326,47,1
2,128,64,42,0,40.0,1.101,24,0	0,137,68,14,148,24.8,0.143,21,0
0,102,52,0,0,25.1,0.078,21,0	0,128,68,19,180,30.5,1.391,25,1
2,146,0,0,0,27.5,0.240,28,1	2,124,68,28,205,32.9,0.875,30,1
10,101,86,37,0,45.6,1.136,38,1	6,80,66,30,0,26.2,0.313,41,0
2,108,62,32,56,25.2,0.128,21,0	0,106,70,37,148,39.4,0.605,22,0
3,122,78,0,0,23.0,0.254,40,0	2,155,74,17,96,26.6,0.433,27,1
1,71,78,50,45,33.2,0.422,21,0	3,113,50,10,85,29.5,0.626,25,0
13,106,70,0,0,34.2,0.251,52,0	7,109,80,31,0,35.9,1.127,43,1
2,100,70,52,57,40.5,0.677,25,0	2,112,68,22,94,34.1,0.315,26,0
7,106,60,24,0,26.5,0.296,29,1	3,99,80,11,64,19.3,0.284,30,0
0,104,64,23,116,27.8,0.454,23,0	3,182,74,0,0,30.5,0.345,29,1
5,114,74,0,0,24.9,0.744,57,0	3,115,66,39,140,38.1,0.150,28,0
2,108,62,10,278,25.3,0.881,22,0	6,194,78,0,0,23.5,0.129,59,1

4,129,60,12,231,27.5,0.527,31,0	5,189,64,33,325,31.2,0.583,29,1
3,112,74,30,0,31.6,0.197,25,1	5,158,70,0,0,29.8,0.207,63,0
0,124,70,20,0,27.4,0.254,36,1	5,103,108,37,0,39.2,0.305,65,0
13,152,90,33,29,26.8,0.731,43,1	4,146,78,0,0,38.5,0.520,67,1
2,112,75,32,0,35.7,0.148,21,0	4,147,74,25,293,34.9,0.385,30,0
1,157,72,21,168,25.6,0.123,24,0	5,99,54,28,83,34.0,0.499,30,0
1,122,64,32,156,35.1,0.692,30,1	6,124,72,0,0,27.6,0.368,29,1
10,179,70,0,0,35.1,0.200,37,0	0,101,64,17,0,21.0,0.252,21,0
2,102,86,36,120,45.5,0.127,23,1	3,81,86,16,66,27.5,0.306,22,0
6,105,70,32,68,30.8,0.122,37,0	1,133,102,28,140,32.8,0.234,45,1
8,118,72,19,0,23.1,1.476,46,0	3,173,82,48,465,38.4,2.137,25,1
2,87,58,16,52,32.7,0.166,25,0	0,118,64,23,89,0.0,1.731,21,0
1,180,0,0,0,43.3,0.282,41,1	0,84,64,22,66,35.8,0.545,21,0
12,106,80,0,0,23.6,0.137,44,0	2,105,58,40,94,34.9,0.225,25,0
1,95,60,18,58,23.9,0.260,22,0	2,122,52,43,158,36.2,0.816,28,0
0,165,76,43,255,47.9,0.259,26,0	12,140,82,43,325,39.2,0.528,58,1
0,117,0,0,0,33.8,0.932,44,0	0,98,82,15,84,25.2,0.299,22,0
5,115,76,0,0,31.2,0.343,44,1	1,87,60,37,75,37.2,0.509,22,0
9,152,78,34,171,34.2,0.893,33,1	4,156,75,0,0,48.3,0.238,32,1
7,178,84,0,0,39.9,0.331,41,1	0,93,100,39,72,43.4,1.021,35,0
1,130,70,13,105,25.9,0.472,22,0	1,107,72,30,82,30.8,0.821,24,0
1,95,74,21,73,25.9,0.673,36,0	0,105,68,22,0,20.0,0.236,22,0
1,0,68,35,0,32.0,0.389,22,0	1,109,60,8,182,25.4,0.947,21,0
5,122,86,0,0,34.7,0.290,33,0	1,90,62,18,59,25.1,1.268,25,0
8,95,72,0,0,36.8,0.485,57,0	1,125,70,24,110,24.3,0.221,25,0
8,126,88,36,108,38.5,0.349,49,0	1,119,54,13,50,22.3,0.205,24,0
1,139,46,19,83,28.7,0.654,22,0	5,116,74,29,0,32.3,0.660,35,1
3,116,0,0,0,23.5,0.187,23,0	8,105,100,36,0,43.3,0.239,45,1
3,99,62,19,74,21.8,0.279,26,0	5,144,82,26,285,32.0,0.452,58,1
5,0,80,32,0,41.0,0.346,37,1	3,100,68,23,81,31.6,0.949,28,0
4,92,80,0,0,42.2,0.237,29,0	1,100,66,29,196,32.0,0.444,42,0
4,137,84,0,0,31.2,0.252,30,0	5,166,76,0,0,45.7,0.340,27,1
3,61,82,28,0,34.4,0.243,46,0	1,131,64,14,415,23.7,0.389,21,0
1,90,62,12,43,27.2,0.580,24,0	4,116,72,12,87,22.1,0.463,37,0
3,90,78,0,0,42.7,0.559,21,0	4,158,78,0,0,32.9,0.803,31,1
9,165,88,0,0,30.4,0.302,49,1	2,127,58,24,275,27.7,1.600,25,0
1,125,50,40,167,33.3,0.962,28,1	3,96,56,34,115,24.7,0.944,39,0
13,129,0,30,0,39.9,0.569,44,1	0,131,66,40,0,34.3,0.196,22,1
12,88,74,40,54,35.3,0.378,48,0	3,82,70,0,0,21.1,0.389,25,0
1,196,76,36,249,36.5,0.875,29,1	3,193,70,31,0,34.9,0.241,25,1

4,95,64,0,0,32.0,0.161,31,1	0,189,104,25,0,34.3,0.435,41,1
6,137,61,0,0,24.2,0.151,55,0	2,83,66,23,50,32.2,0.497,22,0
5,136,84,41,88,35.0,0.286,35,1	4,117,64,27,120,33.2,0.230,24,0
9,72,78,25,0,31.6,0.280,38,0	8,108,70,0,0,30.5,0.955,33,1
5,168,64,0,0,32.9,0.135,41,1	4,117,62,12,0,29.7,0.380,30,1
2,123,48,32,165,42.1,0.520,26,0	0,180,78,63,14,59.4,2.420,25,1
4,115,72,0,0,28.9,0.376,46,1	1,100,72,12,70,25.3,0.658,28,0
0,101,62,0,0,21.9,0.336,25,0	0,95,80,45,92,36.5,0.330,26,0
8,197,74,0,0,25.9,1.191,39,1	0,104,64,37,64,33.6,0.510,22,1
1,172,68,49,579,42.4,0.702,28,1	0,120,74,18,63,30.5,0.285,26,0
6,102,90,39,0,35.7,0.674,28,0	1,82,64,13,95,21.2,0.415,23,0
1,112,72,30,176,34.4,0.528,25,0	2,134,70,0,0,28.9,0.542,23,1
1,143,84,23,310,42.4,1.076,22,0	0,91,68,32,210,39.9,0.381,25,0
1,143,74,22,61,26.2,0.256,21,0	2,119,0,0,0,19.6,0.832,72,0
0,138,60,35,167,34.6,0.534,21,1	2,100,54,28,105,37.8,0.498,24,0
3,173,84,33,474,35.7,0.258,22,1	14,175,62,30,0,33.6,0.212,38,1
1,97,68,21,0,27.2,1.095,22,0	1,135,54,0,0,26.7,0.687,62,0
4,144,82,32,0,38.5,0.554,37,1	5,86,68,28,71,30.2,0.364,24,0
1,83,68,0,0,18.2,0.624,27,0	10,148,84,48,237,37.6,1.001,51,1
3,129,64,29,115,26.4,0.219,28,1	9,134,74,33,60,25.9,0.460,81,0
1,119,88,41,170,45.3,0.507,26,0	9,120,72,22,56,20.8,0.733,48,0
2,94,68,18,76,26.0,0.561,21,0	1,71,62,0,0,21.8,0.416,26,0
0,102,64,46,78,40.6,0.496,21,0	8,74,70,40,49,35.3,0.705,39,0
2,115,64,22,0,30.8,0.421,21,0	5,88,78,30,0,27.6,0.258,37,0
8,151,78,32,210,42.9,0.516,36,1	10,115,98,0,0,24.0,1.022,34,0
4,184,78,39,277,37.0,0.264,31,1	0,124,56,13,105,21.8,0.452,21,0
0,94,0,0,0,0.0,0.256,25,0	0,74,52,10,36,27.8,0.269,22,0
1,181,64,30,180,34.1,0.328,38,1	0,97,64,36,100,36.8,0.600,25,0
0,135,94,46,145,40.6,0.284,26,0	8,120,0,0,0,30.0,0.183,38,1
1,95,82,25,180,35.0,0.233,43,1	6,154,78,41,140,46.1,0.571,27,0
2,99,0,0,0,22.2,0.108,23,0	1,144,82,40,0,41.3,0.607,28,0
3,89,74,16,85,30.4,0.551,38,0	0,137,70,38,0,33.2,0.170,22,0
1,80,74,11,60,30.0,0.527,22,0	0,119,66,27,0,38.8,0.259,22,0
2,139,75,0,0,25.6,0.167,29,0	7,136,90,0,0,29.9,0.210,50,0
1,90,68,8,0,24.5,1.138,36,0	4,114,64,0,0,28.9,0.126,24,0
0,141,0,0,0,42.4,0.205,29,1	0,137,84,27,0,27.3,0.231,59,0
12,140,85,33,0,37.4,0.244,41,0	2,105,80,45,191,33.7,0.711,29,1
5,147,75,0,0,29.9,0.434,28,0	7,114,76,17,110,23.8,0.466,31,0
1,97,70,15,0,18.2,0.147,21,0	8,126,74,38,75,25.9,0.162,39,0
6,107,88,0,0,36.8,0.727,31,0	4,132,86,31,0,28.0,0.419,63,0

3,158,70,30,328,35.5,0.344,35,1	2,68,70,32,66,25.0,0.187,25,0
0,123,88,37,0,35.2,0.197,29,0	3,124,80,33,130,33.2,0.305,26,0
4,85,58,22,49,27.8,0.306,28,0	6,114,0,0,0,0.0,0.189,26,0
0,84,82,31,125,38.2,0.233,23,0	9,130,70,0,0,34.2,0.652,45,1
0,145,0,0,0,44.2,0.630,31,1	3,125,58,0,0,31.6,0.151,24,0
0,135,68,42,250,42.3,0.365,24,1	3,87,60,18,0,21.8,0.444,21,0
1,139,62,41,480,40.7,0.536,21,0	1,97,64,19,82,18.2,0.299,21,0
0,173,78,32,265,46.5,1.159,58,0	3,116,74,15,105,26.3,0.107,24,0
4,99,72,17,0,25.6,0.294,28,0	0,117,66,31,188,30.8,0.493,22,0
8,194,80,0,0,26.1,0.551,67,0	0,111,65,0,0,24.6,0.660,31,0
2,83,65,28,66,36.8,0.629,24,0	2,122,60,18,106,29.8,0.717,22,0
2,89,90,30,0,33.5,0.292,42,0	0,107,76,0,0,45.3,0.686,24,0
4,99,68,38,0,32.8,0.145,33,0	1,86,66,52,65,41.3,0.917,29,0
4,125,70,18,122,28.9,1.144,45,1	6,91,0,0,0,29.8,0.501,31,0
3,80,0,0,0,0.0,0.174,22,0	1,77,56,30,56,33.3,1.251,24,0
6,166,74,0,0,26.6,0.304,66,0	4,132,0,0,0,32.9,0.302,23,1
5,110,68,0,0,26.0,0.292,30,0	0,105,90,0,0,29.6,0.197,46,0
2,81,72,15,76,30.1,0.547,25,0	0,57,60,0,0,21.7,0.735,67,0
7,195,70,33,145,25.1,0.163,55,1	0,127,80,37,210,36.3,0.804,23,0
6,154,74,32,193,29.3,0.839,39,0	3,129,92,49,155,36.4,0.968,32,1
2,117,90,19,71,25.2,0.313,21,0	8,100,74,40,215,39.4,0.661,43,1
3,84,72,32,0,37.2,0.267,28,0	3,128,72,25,190,32.4,0.549,27,1
6,0,68,41,0,39.0,0.727,41,1	10,90,85,32,0,34.9,0.825,56,1
7,94,64,25,79,33.3,0.738,41,0	4,84,90,23,56,39.5,0.159,25,0
3,96,78,39,0,37.3,0.238,40,0	1,88,78,29,76,32.0,0.365,29,0
10,75,82,0,0,33.3,0.263,38,0	8,186,90,35,225,34.5,0.423,37,1
0,180,90,26,90,36.5,0.314,35,1	5,187,76,27,207,43.6,1.034,53,1
1,130,60,23,170,28.6,0.692,21,0	4,131,68,21,166,33.1,0.160,28,0
2,84,50,23,76,30.4,0.968,21,0	1,164,82,43,67,32.8,0.341,50,0
8,120,78,0,0,25.0,0.409,64,0	4,189,110,31,0,28.5,0.680,37,0
12,84,72,31,0,29.7,0.297,46,1	1,116,70,28,0,27.4,0.204,21,0
0,139,62,17,210,22.1,0.207,21,0	3,84,68,30,106,31.9,0.591,25,0
9,91,68,0,0,24.2,0.200,58,0	6,114,88,0,0,27.8,0.247,66,0
2,91,62,0,0,27.3,0.525,22,0	1,88,62,24,44,29.9,0.422,23,0
3,99,54,19,86,25.6,0.154,24,0	1,84,64,23,115,36.9,0.471,28,0
3,163,70,18,105,31.6,0.268,28,1	7,124,70,33,215,25.5,0.161,37,0
9,145,88,34,165,30.3,0.771,53,1	1,97,70,40,0,38.1,0.218,30,0
7,125,86,0,0,37.6,0.304,51,0	8,110,76,0,0,27.8,0.237,58,0
13,76,60,0,0,32.8,0.180,41,0	11,103,68,40,0,46.2,0.126,42,0
6,129,90,7,326,19.6,0.582,60,0	11,85,74,0,0,30.1,0.300,35,0

6,125,76,0,0,33.8,0.121,54,1	1,108,88,19,0,27.1,0.400,24,0
0,198,66,32,274,41.3,0.502,28,1	6,96,0,0,0,23.7,0.190,28,0
1,87,68,34,77,37.6,0.401,24,0	1,124,74,36,0,27.8,0.100,30,0
6,99,60,19,54,26.9,0.497,32,0	7,150,78,29,126,35.2,0.692,54,1
0,91,80,0,0,32.4,0.601,27,0	4,183,0,0,0,28.4,0.212,36,1
2,95,54,14,88,26.1,0.748,22,0	1,124,60,32,0,35.8,0.514,21,0
1,99,72,30,18,38.6,0.412,21,0	1,181,78,42,293,40.0,1.258,22,1
6,92,62,32,126,32.0,0.085,46,0	1,92,62,25,41,19.5,0.482,25,0
4,154,72,29,126,31.3,0.338,37,0	0,152,82,39,272,41.5,0.270,27,0
0,121,66,30,165,34.3,0.203,33,1	1,111,62,13,182,24.0,0.138,23,0
3,78,70,0,0,32.5,0.270,39,0	3,106,54,21,158,30.9,0.292,24,0
2,130,96,0,0,22.6,0.268,21,0	3,174,58,22,194,32.9,0.593,36,1
3,111,58,31,44,29.5,0.430,22,0	7,168,88,42,321,38.2,0.787,40,1
2,98,60,17,120,34.7,0.198,22,0	6,105,80,28,0,32.5,0.878,26,0
1,143,86,30,330,30.1,0.892,23,0	11,138,74,26,144,36.1,0.557,50,1
1,119,44,47,63,35.5,0.280,25,0	3,106,72,0,0,25.8,0.207,27,0
6,108,44,20,130,24.0,0.813,35,0	6,117,96,0,0,28.7,0.157,30,0
2,118,80,0,0,42.9,0.693,21,1	2,68,62,13,15,20.1,0.257,23,0
10,133,68,0,0,27.0,0.245,36,0	9,112,82,24,0,28.2,1.282,50,1
2,197,70,99,0,34.7,0.575,62,1	0,119,0,0,0,32.4,0.141,24,1
0,151,90,46,0,42.1,0.371,21,1	2,112,86,42,160,38.4,0.246,28,0
6,109,60,27,0,25.0,0.206,27,0	2,92,76,20,0,24.2,1.698,28,0
12,121,78,17,0,26.5,0.259,62,0	6,183,94,0,0,40.8,1.461,45,0
8,100,76,0,0,38.7,0.190,42,0	0,94,70,27,115,43.5,0.347,21,0
8,124,76,24,600,28.7,0.687,52,1	2,108,64,0,0,30.8,0.158,21,0
1,93,56,11,0,22.5,0.417,22,0	4,90,88,47,54,37.7,0.362,29,0
8,143,66,0,0,34.9,0.129,41,1	0,125,68,0,0,24.7,0.206,21,0
6,103,66,0,0,24.3,0.249,29,0	0,132,78,0,0,32.4,0.393,21,0
3,176,86,27,156,33.3,1.154,52,1	5,128,80,0,0,34.6,0.144,45,0
0,73,0,0,0,21.1,0.342,25,0	4,94,65,22,0,24.7,0.148,21,0
11,111,84,40,0,46.8,0.925,45,1	7,114,64,0,0,27.4,0.732,34,1
2,112,78,50,140,39.4,0.175,24,0	0,102,78,40,90,34.5,0.238,24,0
3,132,80,0,0,34.4,0.402,44,1	2,111,60,0,0,26.2,0.343,23,0
2,82,52,22,115,28.5,1.699,25,0	1,128,82,17,183,27.5,0.115,22,0
6,123,72,45,230,33.6,0.733,34,0	10,92,62,0,0,25.9,0.167,31,0
0,188,82,14,185,32.0,0.682,22,1	13,104,72,0,0,31.2,0.465,38,1
0,67,76,0,0,45.3,0.194,46,0	5,104,74,0,0,28.8,0.153,48,0
1,89,24,19,25,27.8,0.559,21,0	2,94,76,18,66,31.6,0.649,23,0
1,173,74,0,0,36.8,0.088,38,1	7,97,76,32,91,40.9,0.871,32,1
1,109,38,18,120,23.1,0.407,26,0	1,100,74,12,46,19.5,0.149,28,0

0,102,86,17,105,29.3,0.695,27,0	2,56,56,28,45,24.2,0.332,22,0
4,128,70,0,0,34.3,0.303,24,0	0,162,76,36,0,49.6,0.364,26,1
6,147,80,0,0,29.5,0.178,50,1	0,95,64,39,105,44.6,0.366,22,0
4,90,0,0,0,28.0,0.610,31,0	4,125,80,0,0,32.3,0.536,27,1
3,103,72,30,152,27.6,0.730,27,0	5,136,82,0,0,0.0,0.640,69,0
2,157,74,35,440,39.4,0.134,30,0	2,129,74,26,205,33.2,0.591,25,0
1,167,74,17,144,23.4,0.447,33,1	3,130,64,0,0,23.1,0.314,22,0
0,179,50,36,159,37.8,0.455,22,1	1,107,50,19,0,28.3,0.181,29,0
11,136,84,35,130,28.3,0.260,42,1	1,140,74,26,180,24.1,0.828,23,0
0,107,60,25,0,26.4,0.133,23,0	1,144,82,46,180,46.1,0.335,46,1
1,91,54,25,100,25.2,0.234,23,0	8,107,80,0,0,24.6,0.856,34,0
1,117,60,23,106,33.8,0.466,27,0	13,158,114,0,0,42.3,0.257,44,1
5,123,74,40,77,34.1,0.269,28,0	2,121,70,32,95,39.1,0.886,23,0
2,120,54,0,0,26.8,0.455,27,0	7,129,68,49,125,38.5,0.439,43,1
1,106,70,28,135,34.2,0.142,22,0	2,90,60,0,0,23.5,0.191,25,0
2,155,52,27,540,38.7,0.240,25,1	7,142,90,24,480,30.4,0.128,43,1
2,101,58,35,90,21.8,0.155,22,0	3,169,74,19,125,29.9,0.268,31,1
1,120,80,48,200,38.9,1.162,41,0	0,99,0,0,0,25.0,0.253,22,0
11,127,106,0,0,39.0,0.190,51,0	4,127,88,11,155,34.5,0.598,28,0
3,80,82,31,70,34.2,1.292,27,1	4,118,70,0,0,44.5,0.904,26,0
10,162,84,0,0,27.7,0.182,54,0	2,122,76,27,200,35.9,0.483,26,0
1,199,76,43,0,42.9,1.394,22,1	6,125,78,31,0,27.6,0.565,49,1
8,167,106,46,231,37.6,0.165,43,1	1,168,88,29,0,35.0,0.905,52,1
9,145,80,46,130,37.9,0.637,40,1	2,129,0,0,0,38.5,0.304,41,0
6,115,60,39,0,33.7,0.245,40,1	4,110,76,20,100,28.4,0.118,27,0
1,112,80,45,132,34.8,0.217,24,0	6,80,80,36,0,39.8,0.177,28,0
4,145,82,18,0,32.5,0.235,70,1	10,115,0,0,0,0.0,0.261,30,1
10,111,70,27,0,27.5,0.141,40,1	2,127,46,21,335,34.4,0.176,22,0
6,98,58,33,190,34.0,0.430,43,0	9,164,78,0,0,32.8,0.148,45,1
9,154,78,30,100,30.9,0.164,45,0	2,93,64,32,160,38.0,0.674,23,1
6,165,68,26,168,33.6,0.631,49,0	3,158,64,13,387,31.2,0.295,24,0
1,99,58,10,0,25.4,0.551,21,0	5,126,78,27,22,29.6,0.439,40,0
10,68,106,23,49,35.5,0.285,47,0	10,129,62,36,0,41.2,0.441,38,1
3,123,100,35,240,57.3,0.880,22,0	0,134,58,20,291,26.4,0.352,21,0
8,91,82,0,0,35.6,0.587,68,0	3,102,74,0,0,29.5,0.121,32,0
6,195,70,0,0,30.9,0.328,31,1	7,187,50,33,392,33.9,0.826,34,1
9,156,86,0,0,24.8,0.230,53,1	3,173,78,39,185,33.8,0.970,31,1
0,93,60,0,0,35.3,0.263,25,0	10,94,72,18,0,23.1,0.595,56,0
3,121,52,0,0,36.0,0.127,25,1	1,108,60,46,178,35.5,0.415,24,0
2,101,58,17,265,24.2,0.614,23,0	5,97,76,27,0,35.6,0.378,52,1

4,83,86,19,0,29.3,0.317,34,0
1,114,66,36,200,38.1,0.289,21,0
1,149,68,29,127,29.3,0.349,42,1
5,117,86,30,105,39.1,0.251,42,0
1,111,94,0,0,32.8,0.265,45,0
4,112,78,40,0,39.4,0.236,38,0
1,116,78,29,180,36.1,0.496,25,0
0,141,84,26,0,32.4,0.433,22,0
2,175,88,0,0,22.9,0.326,22,0
2,92,52,0,0,30.1,0.141,22,0
3,130,78,23,79,28.4,0.323,34,1
8,120,86,0,0,28.4,0.259,22,1
2,174,88,37,120,44.5,0.646,24,1
2,106,56,27,165,29.0,0.426,22,0
2,105,75,0,0,23.3,0.560,53,0
4,95,60,32,0,35.4,0.284,28,0
0,126,86,27,120,27.4,0.515,21,0
8,65,72,23,0,32.0,0.600,42,0
2,99,60,17,160,36.6,0.453,21,0
1,102,74,0,0,39.5,0.293,42,1
11,120,80,37,150,42.3,0.785,48,1
3,102,44,20,94,30.8,0.400,26,0
1,109,58,18,116,28.5,0.219,22,0
9,140,94,0,0,32.7,0.734,45,1
13,153,88,37,140,40.6,1.174,39,0
12,100,84,33,105,30.0,0.488,46,0
1,147,94,41,0,49.3,0.358,27,1
1,81,74,41,57,46.3,1.096,32,0
3,187,70,22,200,36.4,0.408,36,1
6,162,62,0,0,24.3,0.178,50,1
4,136,70,0,0,31.2,1.182,22,1
1,121,78,39,74,39.0,0.261,28,0
3,108,62,24,0,26.0,0.223,25,0
0,181,88,44,510,43.3,0.222,26,1
8,154,78,32,0,32.4,0.443,45,1
1,128,88,39,110,36.5,1.057,37,1
7,137,90,41,0,32.0,0.391,39,0
0,123,72,0,0,36.3,0.258,52,1
1,106,76,0,0,37.5,0.197,26,0
6,190,92,0,0,35.5,0.278,66,1
2,88,58,26,16,28.4,0.766,22,0
9,170,74,31,0,44.0,0.403,43,1
9,89,62,0,0,22.5,0.142,33,0
10,101,76,48,180,32.9,0.171,63,0
2,122,70,27,0,36.8,0.340,27,0
5,121,72,23,112,26.2,0.245,30,0
1,126,60,0,0,30.1,0.349,47,1
1,93,70,31,0,30.4,0.315,23,0