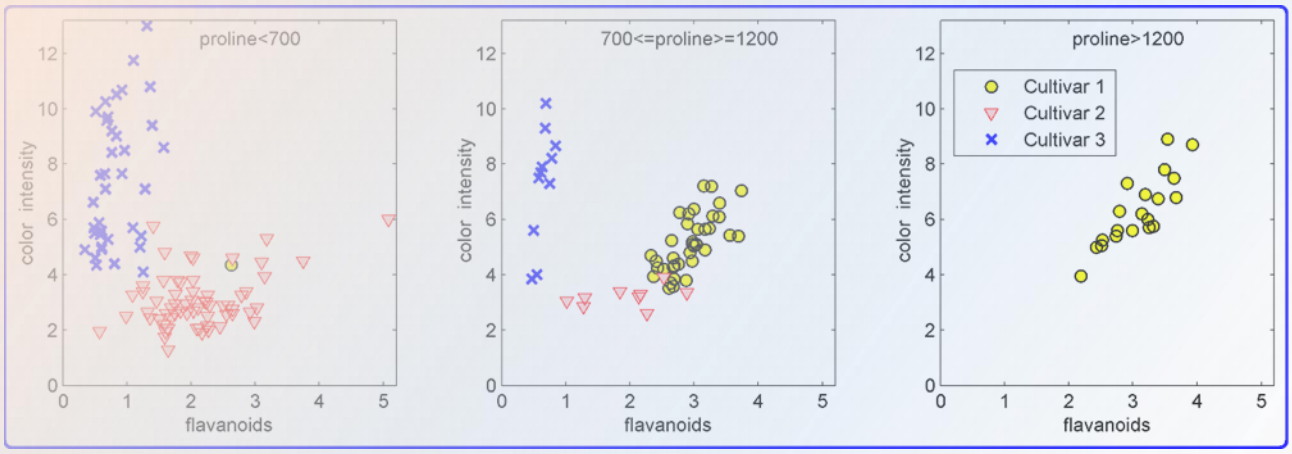
С. Д. Штовба
А. В. Галущак

Ідентифікація багатофакторних залежностей за допомогою баз знань



Лабораторний практикум

УДК 004.82: 004:85
ББК 22.18
Ш92

Рекомендовано до друку Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України та надано гриф електронний навчальний посібник ВНТУ (протокол №7 від 22.12.2015 р.).

Рецензенти:
Р. Н. Квєтний, доктор технічних наук, професор;
А. Я. Кулик, доктор технічних наук, професор;
Д. І. Катєльніков, кандидат технічних наук, доцент.

Штовба С. Д.

Ш92 Ідентифікація багатофакторних залежностей за допомогою баз знань. Лабораторний практикум : електронний навчальний посібник / С. Д. Штовба, А. В. Галущак – Вінниця : ВНТУ, 2015. – 96 с.

У навчальному посібнику наведено теоретичний матеріал з технологій ідентифікації багатофакторних залежностей за допомогою баз знань. Знання представляються деревами рішень та нечіткими правилами. Розглядаються задачі ідентифікації регресійного та класифікаційного типу. Навчальний посібник призначено для студентів напряму підготовки 6.050202 “Автоматизація та комп'ютерно-інтегровані технології”, що вивчають дисципліну “Бази знань та експертні системи”.

IEBN 804-479-000007-31
DOI: 10.13140/RG.2.1.3698.0245

Зміст

Передмова

Ідентифікація - це створення математичних моделей досліджуваних багатофакторних залежностей за результатами спостережень. Ідентифікацію здійснюють у 2 етапи. На першому етапі – етапі структурної ідентифікації формалізують вхідні та вихідні змінні та визначають структуру моделі досліджуваної залежності. На другому етапі – етапі параметричної ідентифікації налаштовують модель, змінюючи її параметри таким чином, щоб поведінка моделі найкращим чином описала експериментальні дані. Цей етап найбільш формалізований і, як правило, зводиться до вирішення задачі оптимізації із мінімізації середньої квадратичної нев'язки для залежності з неперервним виходом чи рівня безпомилковості класифікації для залежності з дискретним виходом.

Сьогодні замовники ідентифікації – медики, економісти, юристи, філологи, політики та інші далекі від математики фахівці все частіше висувають вимоги не лише щодо точності синтезованих моделей. Їм потрібно розуміти чому модель радить те чи інше рішення. Вони не звикли довіряти прийняття важливих рішень набору чисел без змістовної інтерпретації, наприклад, параметрам штучної нейронної мережі. Тому нині все більшу зацікавленість викликають технології ідентифікації, які продукують інтерпретабельні моделі. Одними з найбільш популярних серед них є моделі на основі баз знань, що містять множину продукційних правил <Якщо - тоді>. На навчання технологіям ідентифікації залежностей за допомогою саме баз знань і спрямовано цей посібник. В ньому розглядаються 2 способи подання знань: чіткими правилами у форматі дерев рішень та нечіткими правилами у форматі баз знань Мамдані та нечіткого класифікатора.

Незважаючи на велику кількість друкованих праць та електронних ресурсів з теорії ідентифікації сьогодні недостатньо систематизовано наукові знання, які були б корисними під час вирішення реальних задач з використанням сучасних інформаційних технологій. У переважній більшості джерел з ідентифікації увага зосереджена на строгості математичних викладок. При цьому інформація про сучасні математичні пакети, в яких

автоматизовано більшість алгоритмів ідентифікації, надається фрагментарно, а часто і взагалі відсутня. Вивчаючи такі книги студенти можуть накопичити гарні теоретичні знання, але лише одиниці здатні самостійно перетворити їх в навички розв'язання практичних задач ідентифікації.

Під час роботи над посібником автори ставили за мету не лише описати теоретичні положення, але і сформувати уміння вирішувати практичні задачі ідентифікації з допомогою сучасного інструментарію – програмної системи MATLAB 7. Посібник складається з трьох лабораторних робіт, кожна з яких є невеликим повноцінним науковим дослідженням. Ще декілька років тому кожна з них за трудомісткістю тягнула на курсову роботу. Але сьогодні програмні пакети забезпечили автоматизацію більшості математичних розрахунків та інших рутинних процедур. Відповідно, студенти можуть зосередитися на аналітичній роботі – постановці задачі, виборі технології її розв'язання та інтерпретації результатів.

Особливістю лабораторних робіт є проведення дослідження на реальних задачах із репозитарію машинного навчання Каліфорнійського університету в Ірвіні. Ці задачі фактично стали міжнародним стандартом для тестування нових методів ідентифікації. Тому студентам пропонується порівняти власні результати ідентифікації залежності і здобутки конкурентних досліджень, які опублікувано в релевантних наукових статтях.

Передбачається, що студенти володіють ключовими навиками роботи в програмному середовищі MATLAB 7. Для новачків рекомендуємо компактний посібник [7]. При підготовці посібника використано таку літературу: для лабораторної роботи №1 – [1–5, 8, 10, 12, 17, 24]; для лабораторної роботи №2 – [5, 6, 8, 10, 11, 14–19, 21]; для лабораторної роботи №3 – [6, 8–11, 13, 18–21, 23, 24]. Ця література рекомендується для поглибленого вивчення матеріалу. Уся література доступна в Інтернеті.

За кожною лабораторною роботою необхідно оформити звіт, в якому окрім постановки задачі та результатів виконання завдань необхідно навести лістинги усіх програм та основні графічні вікна. Особливу увагу слід приділити висновкам та інтерпретації результатів.

Публікація містить результати досліджень, проведених при грантовій підтримці Державного фонду фундаментальних досліджень за конкурсним проектом №Ф62/201-2015.

Лабораторна робота №1

Екстракція з експериментальних даних класифікатора у формі дерева рішень

Мета – віднайти дерево рішень оптимального розміру, яке забезпечує найкращу точність класифікації для заданого набору експериментальних даних.

Теоретичні відомості

Розглядається задача класифікації (рис. 1), тобто віднесення об'єкта з ознаками (атрибутами) $X=(x_1, x_2, ..., x_n)$ до одного із класів $\{l_1, l_2, ..., l_m\}$. З математичної точки зору класифікація – це відображення виду $X=(x_1, x_2, ..., x_n) \to y \in \{l_1, l_2, ..., l_m\}$. До класифікації зводяться різноманітні задачі прийняття рішень та розпізнавання образів в інженерному проектуванні, військовій справі, менеджменті, політиці, спорті, в медичній, технічній і економічній діагностиках тощо.

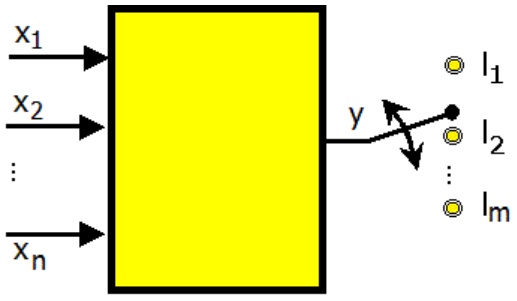


Рисунок 1 – Задача класифікації

Класифікатор будемо створювати лише за вибіркою експериментальних даних:

$$(X_r, y_r), \quad r=\overline{1, M}, \tag{1}$$

де $X = (x_{r1}, x_{r2}, ..., x_{rn})$;

$y_r \in \{l_1, l_2, ..., l_m\}$;

M – довжина вибірки.

Якість класифікатора оцінюють за критеріями складності, вартості, інтерпретуємості та точності. *Складність* класифікатора визначають за кількістю елементарних операцій, які необхідно виконати для прийняття рішення. *Вартість* визначається витратами на збір початкових даних, необхідних для прийняття рішення. Кожен додатковий атрибут, який використовується для прийняття рішень, вимагає витрат деяких ресурсів – наприклад, проведення лабораторних досліджень для медичної діагностики. Тому, намагаються розробити класифікатор таким чином, щоб дорогі атрибути використовувати рідко. *Інтерпретуємість* пов'язують з наочністю моделі. Вона віддзеркалює наскільки процес прийняття рішень є зрозумілим фахівцям з предметної області, які не мають спеціальної кібернетичної підготовки. Точність класифікатора F зазвичай визначають за частотою помилок:

$$MCR(F) = \frac{\sum\limits_{j=\overline{1, M}} \Delta_j}{M},$$

де $\Delta_j = \begin{cases} 1, & якщо \ \ y_j \neq F(X_j) \\ 0, & якщо \ \ y_j = F(X_j) \end{cases}$.

У вибірці (1) вхідні змінні можуть приймати значення з різних шкал. Найчастіше зустрічаються такі типи шкал даних:

числова, наприклад, діапазон $[-35, 40]$ для оцінювання температури повітря;

категоріальна або номінальна, наприклад, множина {чоловіча, жіноча} для опису статі людини або множина {червоний, жовтий, зелений} для опису кольору;

порядкова, наприклад, множина {незадовільно, задовільно, добре, відмінно} для оцінювання знань студента.

Над даними з числової шкали можна виконувати арифметичні операції. Для цієї шкали існують відношення еквівалентності та порядку, тобто над числовими даними можна виконати логічні операції дорівнює, більше та

менше. Для категоріальної шкали допустиме лише одне відношення – відношення еквівалентності. Над даними з порядкової шкали можна виконувати логічні операції дорівнює, більше або менше, але арифметичні операції є недопустимі, навіть якщо для запису порядкових оцінок використовуються числа.

Класифікатори бувають параметричними та непараметричними. В параметричних класифікаторах наперед визначена модель прийняття рішень, наприклад, деяке рівняння кривої розділу класів в просторі вхідних атрибутів. Синтез класифікатора полягає в знаходженні таких параметрів цієї моделі, які забезпечують найкращу якість прийняття рішень. В непараметричних класифікаторах структура моделі визначається не суб'єктивно, а за деяким алгоритмом аналізу вибірки (1). В лабораторній роботі досліджується один із непараметричних класифікаторів – дерево рішень.

Дерево рішень (*decision tree*) є однією з найпопулярніших моделей класифікації. Воно являє собою ієрархічний набір правил "Якщо – тоді – інакше" у формі орієнтованого дерева. Зазвичай використовують бінарне дерево рішень, з кожної нетермінальної вершини якого виходять 2 дуги (рис. 2). Корінь дерева є фіктивним; він позначає початок класифікації. Листки дерева відповідають класам рішень, причому одному класу може належати кілька листків. Проміжні вершини відповідають логічним умовам, за якими аналізуються атрибути – ознаки об'єкта класифікації. Щоб визначити клас належності деякого об'єкту потрібно в нетермінальних вершинах дерева відповісти на питання типу: "Значення ознаки *x* менше *A*?" або "Значення ознаки *x* дорівнює *A* або *B*". Якщо відповідь "так", здійснюється перехід до лівої вершини наступного рівня дерева, якщо "ні"– до правої вершини. Рух по дереву продовжується до тих пір, поки не потрапимо на один із його листків.

Для синтезу дерев рішень найчастіше використовуються жадібні алгоритми. Дерево рішень будують від кореня до листків. На кожній ітерації 1 листок дерева розщеплюють – перетворюють на проміжну вершину з двома новими листками. При появі нових листків їм потрібно поставити мітки, тобто вказати належність до того чи іншого класу. Очевидно, що 2 сусідні листки повинні мати різні мітки. Мітка відповідає тому класу, об'єкти якого домінують на листку. Наприклад, на листок v_1 з навчальної вибірки потрапило 10 об'єктів класу l_1, 5 об'єктів класу l_2, та 1 об'єкт

класу l_3. А на листок v_2 потрапило 4 об'єктів класу l_1, 2 об'єкти класу l_2, та 8 об'єктів класу l_3. Тоді листок v_1 помітимо як l_1, а v_2 листок як l_3.

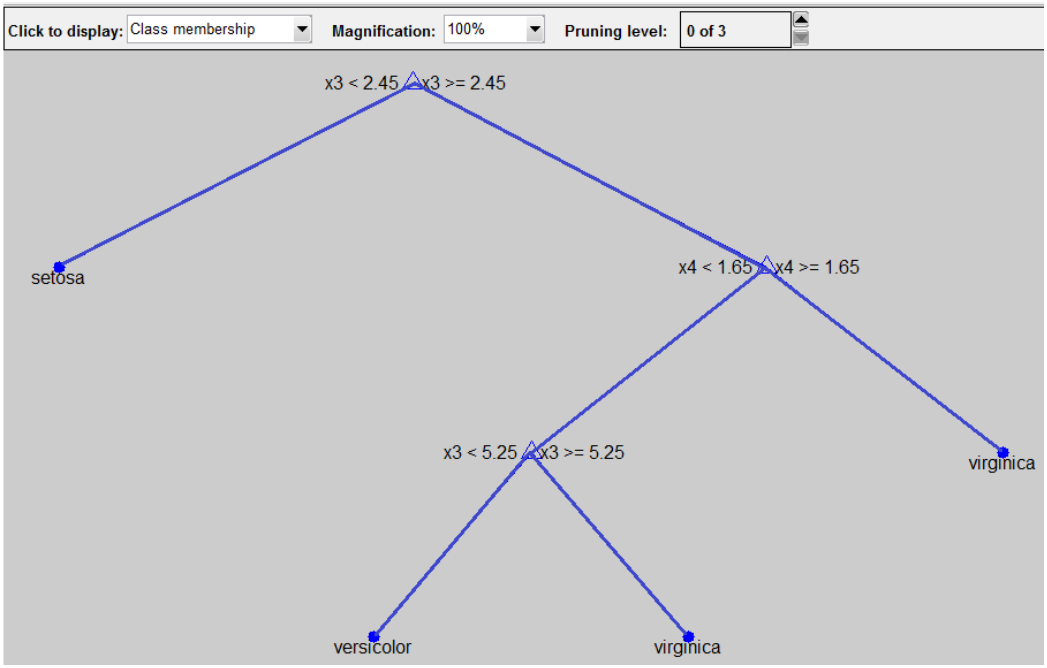


Рисунок 2 – Дерево рішень для класифікації ірисів в форматі MATLAB 7

Під час екстракції дерева рішень логічна умова в новій проміжній вершині формується так, щоб забезпечити екстремальне значення деякого локального критерія ефективності. В логічній умові може фігурувати лише одна із n ознак $x_1, x_2, ..., x_n$. Для кожної із ознак генерується скінченна множина варіантів логічних умов.

Якщо ознака x_i, $i = \overline{1,n}$ задана на числовій шкалі, тоді для формування правої частини логічної умови $x_i < a$ відсортуємо значення x_i для об'єктів навчальної вибірки, які потрапили в аналізовану проміжну вершину. Позначимо відсортований ряд цих значень через $(a_1, a_2, ..., a_k)$. Тоді $a \in \left\{ \frac{a_1 + a_2}{2}, \frac{a_2 + a_3}{2}, ..., \frac{a_{k-1} + a_k}{2} \right\}$. Звідси, мінімальна кількість варіантів умови дорівнює 1, що відповідає випадку двох можливих значень змінної x_i. За одного можливого значення змінної x_i розщеплення за цією ознакою буде неможливим. Максимальна кількість можливих умов дорівнює $M - 1$, що може бути тільки при формуванні першої проміжної вершини. При цьому в вибірці (1) усі значення змінної x_i мають бути

унікальними, тобто $k = M$. Максимальна кількість варіантів логічної умови для однієї вершини дорівнює $n \cdot (M - 1)$.

Якщо ознака x_i задана на порядковій шкалі, тоді для формування правої частини логічної умови $x_i < a$ відсортуємо за зростанням значення x_i для об'єктів навчальної вибірки, які потрапили в проміжну вершину. Позначимо відсортований ряд цих значень через $(a_1, a_2, ..., a_k)$. Тоді $a \in \{a_2, a_3, ..., a_k\}$.

Якщо ознака x_i задана на категоріальній шкалі, тоді сформуємо множину B значень x_i для об'єктів навчальної вибірки, що потрапили в проміжну вершину. Логічну умову в цій проміжній вершині запишемо так: $x_i \in A$, де $A \subset B$, причому $0 < |A| < |B|$.

Вибір вершини для розщеплення та логічної умови в ній, залежить від алгоритму синтезу дерева рішень. На сьогодні запропоновано кілька ефективних жадібних алгоритмів синтезу дерев рішень, найбільш популярними серед яких є CART та C4.5. Алгоритм CART реалізовано в пакеті Statistics Toolbox програмної системи MATLAB 7, а алгоритм C4.5 – в програмному продукті See5.

В алгоритмі CART (*Classification and Regression Tree*) правило розщеплення побудовано за принципом мінімізації індекса Джині. Індекс Джині або коефіцієнт Джині походить з соціології, де він є індикатором нерівномірності розподілу доходів або розподілу багатства, тобто показує наскільки сильно суспільство розшаровано на багатих та бідних. В алгоритмі CART індекс Джині використовується для оцінювання якості умови в проміжній вершині дерева. Чим більше розшарування між класами в листках, що пов'язані з цією вершиною, тим краща її якість. Індекс Джині вершини v, з якою пов'язані листки v_1 та v_2 розраховується так:

$$G(v) = \frac{N_1}{N}\left(1 - \sum_{j=\overline{1,m}} p_{1j}^2\right) + \frac{N_2}{N}\left(1 - \sum_{j=\overline{1,m}} p_{2j}^2\right), \qquad (2)$$

де N_1 та N_2 кількість об'єктів навчальної вибірки, які потрапили на листки v_1 та v_2;

N – кількість об’єктів навчальної вибірки, які потрапили на проміжну вершину v, $N = N_1 + N_2$;

p_{1j} – частота потрапляння на листок v_1 об’єктів з класу l_j, $j = \overline{1, m}$;

p_{2j} – частота потрапляння на листок v_2 об’єктів з класу l_j, $j = \overline{1, m}$.

В алгоритмі С4.5 правило розщеплення побудовано за принципом мінімізації ентропії. Ентропія – це міра безладу. Абсолютний порядок на листку дерева рішень буде, коли на нього потрапили об’єкти лише одного класу. Абсолютний безлад буде, коли на листку всі класи представлені рівномірно. Ідея алгоритму С4.5 полягає у виборі варіанту логічної умови, який найсильніше покращує порядок, тобто найбільше зменшує ентропію. Для цього спочатку розрахуємо ентропію навчальної вибірки:

$$I = -\sum_{j=\overline{1, m}} p_j \cdot \log_2(p_j),$$

де p_j – частота об’єктів класу l_j у вибірці, $j = \overline{1, m}$;

Далі розрахуємо ентропію кожного варіанту логічної умови:

$$I(v) = -\frac{N_1}{N} \sum_{j=\overline{1, m}} p_{1j} \cdot \log_2(p_{1j}) - \frac{N_2}{N} \sum_{j=\overline{1, m}} p_{2j} \cdot \log_2(p_{2j}).$$

Кращою буде логічна умова, яка забезпечує максимальне зменшення ентропії:

$$\Delta I(v) = I - I_v \to \max.$$

Правило розщеплення за індексом Джині формує листки дерева таким чином, щоб ізолювати об’єкти найпоширенішого класу. За ентропійним правилом обирається логічна умова, яка забезпечує приблизно однакову кількість об’єктів на сусідніх листках.

Приклад. В вершину v потрапило $N = 200$ об’єктів, серед яких 100 належать класу l_1 і 100 – класу l_2. Можливі 2 варіанти логічної умови в вершині v, розподіл класів за якими наведено в табл. 1.

Таблиця 1 – Розподіл класів по листкам

Варіант логічної умови	v_1		v_2	
	l_1	l_2	l_1	l_2
I	5	50	95	50
II	5	90	95	10

Вершина v є абсолютно забрудненою – співвідношення між об'єктами різних класів складає 1:1. За першим варіантом логічної умови отримуємо в вершині v_1 рівень забруднення 1:10 та в вершині v_2 – приблизно 2:1. За другим варіантом логічної умови отримуємо рівень забруднення 1:18 в вершині v_1 та 9.5:1 в вершині v_2. Очевидно, що другий варіант є кращим. Порівняємо цей висновок з результатами прийняття рішення за алгоритмами CART та C4.5.

В алгоритмі CART вибір логічної умови для розщеплення вершини здійснюється за індексом Джині.

Індекс Джині першого варіанта дорівнює:

$$G(v_{\mathrm{I}}) = \frac{55}{200}\left(1-\left(\frac{5}{55}\right)^2-\left(\frac{50}{55}\right)^2\right)+\frac{145}{200}\left(1-\left(\frac{95}{145}\right)^2-\left(\frac{50}{145}\right)^2\right)=$$
$$= \frac{55}{200}\cdot 0.1653+\frac{145}{200}\cdot 0.4518 = 0.373.$$

Індекс Джині другого варіанта дорівнює:

$$G(v_{\mathrm{II}}) = \frac{95}{200}\left(1-\left(\frac{5}{95}\right)^2-\left(\frac{90}{95}\right)^2\right)+\frac{105}{200}\left(1-\left(\frac{95}{105}\right)^2-\left(\frac{10}{105}\right)^2\right)=$$
$$= \frac{95}{200}\cdot 0.0997+\frac{105}{200}\cdot 0.1723 = 0.1378.$$

З розрахунків видно, що $G(v_{\mathrm{II}}) < G(v_{\mathrm{I}})$, тому, як і очікувалось, другий варіант є кращим.

В алгоритмі C4.5 вибір умови для розщеплення вершини здійснюється за ентропійним правилом. Спочатку розрахуємо початкову ентропію:

$$I = -\frac{100}{200}\log_2\left(\frac{100}{200}\right) - \frac{100}{200}\log_2\left(\frac{100}{200}\right) = -0.5\cdot(-1) - 0.5\cdot(-1) = 1.$$

Ентропія для першого варіанта логічної умови дорівнює:

$$\begin{aligned} I_{v_1} &= -\frac{55}{200}\left(\frac{5}{55}\log_2\left(\frac{5}{55}\right) + \frac{50}{55}\log_2\left(\frac{50}{55}\right)\right) - \\ &- \frac{145}{200}\left(\frac{95}{145}\log_2\left(\frac{95}{145}\right) + \frac{50}{145}\log_2\left(\frac{50}{145}\right)\right) = \\ &= -0.275(0.09\cdot(-3.4739) + 0.91\cdot(-0.1361)) - \\ &- 0.725(0.66\cdot(-0.5995) + 0.34\cdot(-1.5564)) = \\ &= 0.275\cdot 0.4366 + 0.725\cdot 0.9249 = 0.7906. \end{aligned}$$

Ентропія для другого варіанта логічної умови дорівнює:

$$\begin{aligned} I_{v_{\text{II}}} &= -\frac{95}{200}\left(\frac{5}{95}\log_2\left(\frac{5}{95}\right) + \frac{90}{95}\log_2\left(\frac{90}{95}\right)\right) - \\ &- \frac{105}{200}\left(\frac{95}{105}\log_2\left(\frac{95}{105}\right) + \frac{10}{105}\log_2\left(\frac{10}{105}\right)\right) = \\ &= -0.475(0.05\cdot(-4.3219) + 0.95\cdot(-0.0740)) - \\ &- 0.525(0.9\cdot(-0.152) + 0.1\cdot(-3.3219)) = \\ &= 0.475\cdot 0.2864 + 0.525\cdot 0.4690 = 0.3823. \end{aligned}$$

Зменшення ентропії становить:

$$\Delta I(v_{\text{I}}) = 1 - 0.7906 = 0.2094;$$

$$\Delta I(v_{\text{II}}) = 1 - 0.3823 = 0.6177.$$

Видно, що $\Delta I(v_{\text{II}}) > \Delta I(v_{\text{I}})$, тому, як і очікувалось, другий варіант є кращим.

Алгоритм синтезу дерева рішень збільшує кількість проміжних вершин до тих пір, поки не почне виконуватися один із критеріїв зупинки. Сьогодні обґрунтовано щонайменше 5 критеріїв зупинки, з яких на практиці переважно використовуються лише 3. Перший критерій – досягнення максимальної глибини дерева, яка розраховується як кількість проміжних вершин від кореня до листка. Другий критерій – мінімальна кількість об’єктів навчальної вибірки, які потрапляють на один листок

дерева. Наприклад, якщо потенційна проміжна вершина дерева рішень породжує листки, в один із яких попадає менше 10 об'єктів, то таке розбиття вважається недоцільним і розщеплення не фіксується. Відповідно за цим напрямком припиняється подальше розгалуження, і потенційна проміжна вершина залишається листком. Третій критерій – неможливість подальшого розщеплення листка, наприклад, коли в нього потрапили об'єкти лише одного класу або об'єкти з однаковими значенням усіх ознак.

Синтезоване за жадібними алгоритмами дерево рішень як правило виходить громіздким та заскладним. Для такого дерева частота помилкової класифікації на тестовій вибірці зазвичай значно більша, ніж на навчальній. Для покращення класифікатора дерево рішень підрізають. Підрізання здійснюють або за рівнями – зменшуючи глибину дерева, або за вершинами - видаляючи окремі листки.

Типові залежності частоти помилкової класифікації (*MCR*) від розміру дерева рішень (*Complexity*) наведені на рис. 3. На навчальній вибірці залежність є монотонно спадною, тоді як на тестовій вибірці спостерігається екстремум. За принципом зовнішнього доповнення моделлю класифікації слід обрати дерево рішень з мінімальною кількістю помилок на тестовій вибірці. На рис. 3 це дерево позначено літерою "A".

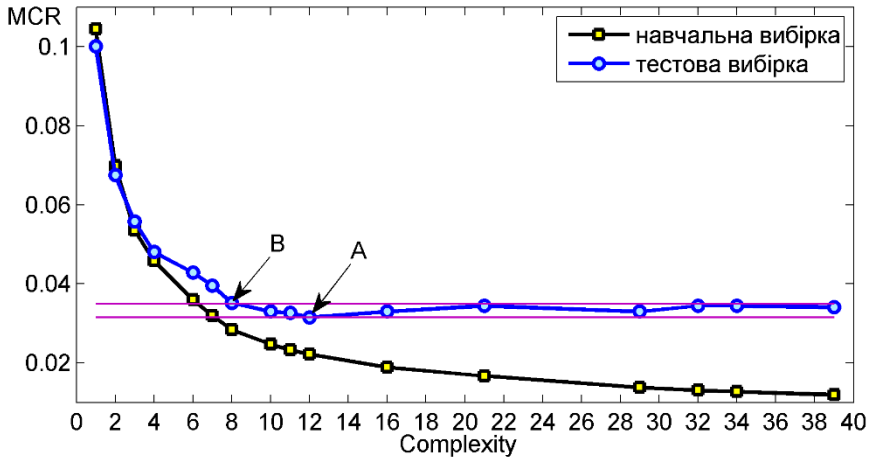


Рисунок 3 – Залежність частоти помилкової класифікації від кількості листків дерева для задачі Page Blocks Classification

За рис. 3 кращою моделлю класифікації відповідно до алгоритму CART буде обрано не дерево "A", а більш просте дерево "B". Обґрунтування такого вибору полягає в тому, що залежності на рис. 3 не є абсолютно достовірними, оскільки значення атрибутів містять деякі шуми. Крім того,

криві на рис. 3 чутливі до способу розбиття експериментальних даних на навчальну та тестову вибірки. За іншого розбиття частота помилок класифікації могла б бути іншою. Відповідно на рис. 3 частоти помилкової класифікації оцінено з деякою похибкою. Ця похибка розраховується так:

$$\Delta = \sqrt{\frac{MCR \cdot (1 - MCR)}{M_{test}}}, \qquad (3)$$

де M_{test} – кількість об'єктів тестової вибірки.

За формулою (3) на рис. 3 проведено Δ-коридор, в межах якого класифікатори є статистично нерозрізненими за частотою помилки. Звичайно, серед моделей однакової безпомилковості слід обирати найпростішу, якою є класифікатор "B" на рис. 3. Початкове та оптимальне дерева рішень наведені на рис. 4 та 5. Аналізуючи ці класифікатори бачимо, що оптимальне дерево не тільки забезпечує кращу безпомилковість, але і має в 5 разів менше логічних умов. Крім того, для класифікації за оптимальним деревом рішень потрібні значення лише 4 ознак, тоді як за початковим деревом – 9. Таким чином, оптимальне дерево краще за критеріями точності, складності, вартості та інтерпретуємості.

Обирати дерево рішень за критерієм *MCR* доцільно у випадку однакової вартості помилок класифікації різних типів. На практиці зустрічаються задачі коли вартості помилок різних типів суттєво різняться. Наприклад, вартість помилок першого роду (пропуск цілі) в кілька разів більша за вартість помилок другого роду (хибна тривога). Інший приклад. Під час оцінювання знань студентів за шкалою {"2", "3", "4", "5"} зустрічаються 12 типів помилок. Вартість помилок типу $5 \to 2$ значно вища за помилки типу $3 \to 4$ чи $4 \to 3$.

Для врахування помилок різних типів використовують 2 підходи. За першим підходом в постановку задачі навчання вводять обмеження на допустимі рівні помилок найбільш небезпечних типів. У випадку лише двох типів помилок у відповідності до леми Неймана–Пірсона мінімізують кількість помилок другого роду при обмежені на рівень помилок першого роду. За другим підходом мінімізують ризик – сумарні збитки від помилкової класифікації. В цьому випадку вважають відомими вартості помилок кожного типу, які зазвичай задають платіжною матрицею.

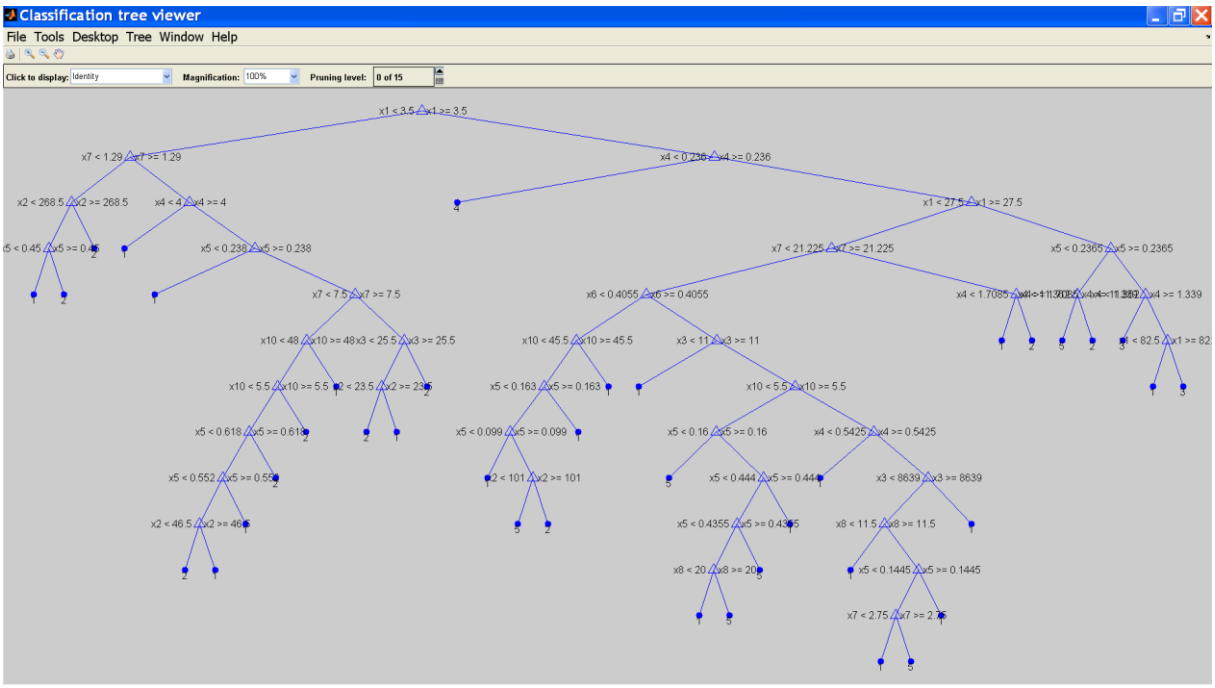


Рисунок 4 – Початкове дерево рішень для задачі Page Blocks Classification в головному вікні модуля Classification tree viewer

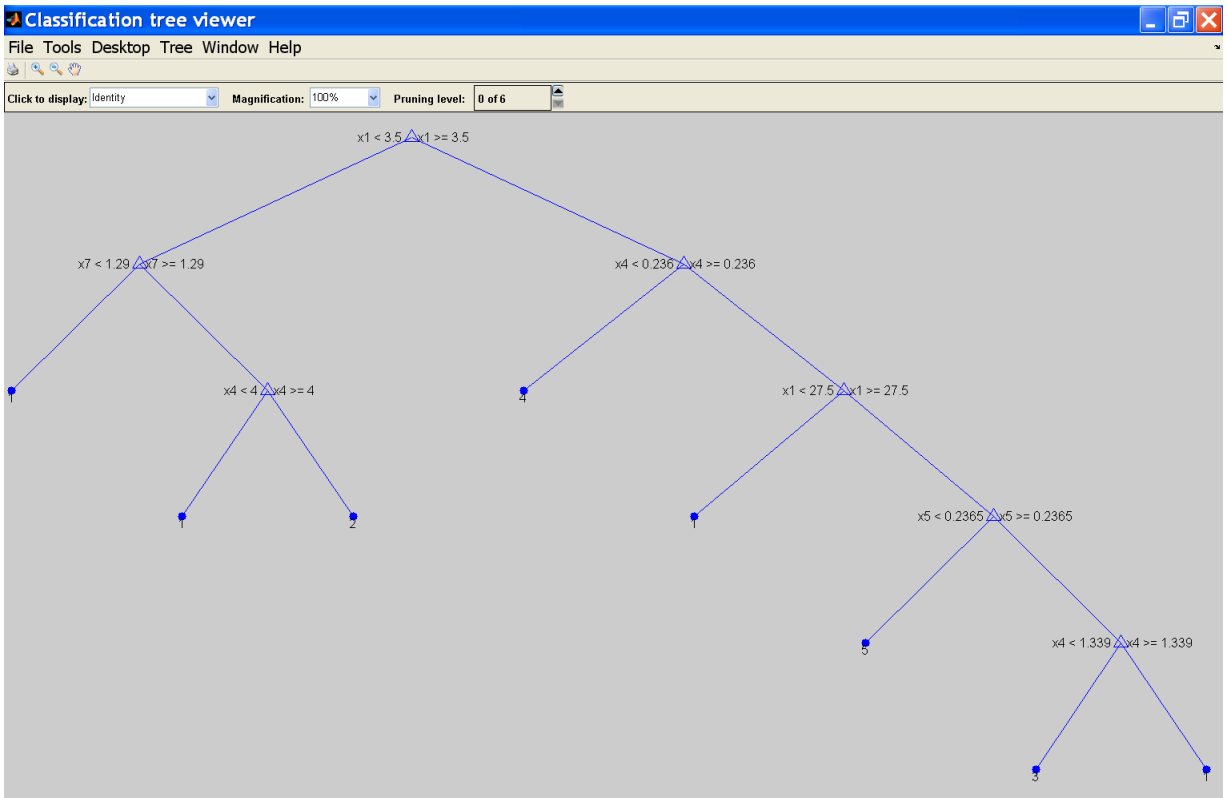


Рисунок 5 – Оптимальне дерево рішень для задачі Page Blocks Classification в головному вікні модуля Classification tree viewer

Платіжною називається наступна квадратна матриця:

$$\mathbf{C} = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1m} \\ c_{21} & c_{22} & \dots & c_{2m} \\ \vdots & & & \\ c_{m1} & c_{m2} & \dots & c_{mm} \end{bmatrix}, \qquad (4)$$

де c_{ij} – ціна помилки типу $l_i \to l_j$, коли замість вірного рішення l_i під час класифікації помилково обрано рішення l_j, $i = \overline{1, m}$, $j = \overline{1, m}$. Усі елементи головної діагоналі матриці (4) дорівнюють нулю, $c_{ii} = 1$, $i = \overline{1, m}$, що вказує на відсутність штрафу за правильну класифікацію.

Для розрахунку ризику окрім платіжної матриці слід знати і матрицю сплутувань, яка записується таким чином:

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & \dots & p_{1m} \\ p_{21} & p_{22} & \dots & p_{2m} \\ \vdots & & & \\ p_{m1} & p_{m2} & \dots & p_{mm} \end{bmatrix}, \qquad (5)$$

де p_{ij} – ймовірність події $l_i \to l_j$, $i = \overline{1, m}$, $j = \overline{1, m}$.

За відомих матриць (4) та (5) ризик розраховується таким чином:

$$R = \sum_{i=\overline{1, m}} \sum_{j=\overline{1, m}} p_{ij} \cdot c_{ij}.$$

В багатьох практичних задачах експериментальні дані містять пропуски – тобто у об’єкта класифікації відсутнє значення деякої ознаки. Як діяти в такому випадку?

Перший варіант – вилучити об’єкти з пропусками з вибірки даних. Але так можна неприпустимо скоротити обсяг початкової інформації, що призведе до зниження достовірності моделі. Крім того, можливий “зсув” моделі, коли об’єкти з пропусками та без пропусків знаходяться в різних зонах факторного простору. Наприклад, в деяких містах опитування здійснювалося за старими анкетами, а в деяких - за новими, з додатковим запитанням. Відповідно, записи з старих анкет містять пропуски. Але вилучивши їх, ми суттєво спотворимо вибірку даних.

Другий варіант – вилучити з вибірки ознаки, що містить пропуски. Але тут можлива ситуація, коли за кожною ознакою хоча б в одному рядку вибірки даних є пропуски. Тоді виходить, що треба вилучити усі ознаки. Можлива й інша ситуація. Зібрана велика вибірка, наприклад, результати анкетування 10 000 осіб. І в одній анкеті респондент не відповів на якесь питання, наприклад, не вказав свій вік. І що, через 1 пропуск ігнорувати відповіді 9 999 респондентів? Очевидно, що шкода втрачати такі дані лише через 1 пропуск.

Третій варіант – замінити пропуски на значення “невідомо”, тобто розширити шкалу вимірювань. Такий варіант можна застосувати, якщо пропуск стосується категоріальної ознаки.

Четвертий варіант – замінити пропуски ймовірнісним розподілом. Цей розподіл будується за наявними в вибірці значеннями проблемної ознаки. Сучасні алгоритми здатні синтезувати дерева рішень з урахуванням таких ймовірнісних розподілів випадкових величин.

Популярність дерев рішень для ідентифікації реальних залежностей обумовлена: 1) наочністю і зрозумілістю моделі; 2) простотою процедури класифікації; 3) можливістю використання як числових, так і порядкових та категоріальних атрибутів. Перевагою також є те, що синтез та оптимізація дерев рішень відбуваються швидко навіть для великих вибірок даних. При цьому, досліднику не потрібно вказувати, які атрибути є інформативні, а які – ні. Формування переліку інформативних атрибутів здійснюється покроково під час синтезу дерева рішень.

Особливість дерев рішень полягає в тому, що границі розділу класів є прямокутними. Тому для складних задач з іншими границями розділу класів застосування дерев рішень є недоцільним. Одним із шляхів підвищення достовірності є застосування не одного класифікатора, а ансамблю (колективу) класифікаторів, за яким можна реалізувати довільні границі розділу класів. Є багато підходів до колективного прийняття рішень, наприклад, створення моделі, за якою обирається класифікатор для кожної області факторного простору. Тобто для кожного класифікатора визначається область його компетенції. Інший підхід полягає в класифікації одного і того ж об’єкту одночасно кількома класифікаторами. Кожен класифікатор видає своє рішення, які потім агрегують.

Найпростішим способом агрегування є вибір рішення, за який проголосувало найбільше класифікаторів.

Одним із найбільш ефективних методів синтезу ансамблю класифікаторів є бустинг дерев рішень. Бустинг (від англ. *boosting* – форсування, підсилення) – це метод синтезу високоточного класифікатора з низькоточних простих класифікаторів. За одну ітерацію бустингу в ансамбль додається 1 класифікатор, який забезпечує найбільше зростання безпомилковості. При цьому попередні класифікатори не видаляються. Таким чином, бустинг має ознаки жадібного алгоритму. Дерево рішень на кожній ітерації синтезується з деякої підмножині навчальної вибірки за типовим алгоритмом наприклад, CART чи C4.5. Ключовою особливістю бустингу є адаптивність формування навчальної підвибірки на кожній ітерації. Підвибірка формується випадково, але так, щоб більше шансів потрапити в неї мали проблемні об’єкти. Під проблемними розуміються об’єкти, які частіше за інші хибно класифікуються деревами рішень з поточного ансамблю. Таким чином, кожне нове дерево ансамблю намагається виправити помилки попередніх. Бустинг часто реалізують алгоритмами AdaBoost та arc-x4.

Під час вирішення практичних задач виявлено, що ефект перенавчання ансамблю класифікаторів, синтезованих за бустингом дерев рішень, спостерігається рідко. Тобто, зі збільшенням кількості дерев в ансамблі зменшується кількість помилкових рішень як для навчальної, так і для тестової вибірок. Збільшуючи розмір ансамблю можна досягти нульової частоти помилок на навчальній вибірці. Якщо продовжити додавати в ансамбль нові дерева, тоді ще протягом кількох ітерацій частота помилок на тестовій вибірці може спадати. Але тут слід пам’ятати, що безпомилковість класифікатора – це не єдиний критерій якості моделі прийняття рішень. При збільшенні ансамблю погіршуються інші критерії – зростає обчислювальна складність та втрачається наочність. Крім того, в модель додаються нові вхідні змінні, яких не було в попередніх деревах рішень. Відповідно, під час застосування класифікатора на практиці зростають витрати на отримання початкових даних через встановлення нових вимірювальних приладів в задачах технічної діагностики чи проведення додаткових лабораторних обстежень пацієнтів в задачах медичної діагностики.

Завдання на лабораторну роботу

В лабораторній роботі досліджується вплив розміру дерева рішень на безпомилковість класифікації об’єктів для однієї із задач репозитарію автоматичного навчання Каліфорнійського університету в Ірвіні [22]. Лабораторна робота полягає у виконанні таких завдань.

1. Сформулювати змістовну постановку задачі згідно варіанту з табл. 0.
2. Розбити дані на навчальну та тестову вибірки та перевірити їх репрезентативність.
3. Синтезувати дерево рішень для здійснення класифікації об’єктів для задачі згідно варіанту з табл. 0.
4. Побудувати залежність частоти помилок класифікації (або ризику, якщо задана платіжна матриця) від розміру дерева рішень.
5. Обґрунтувати оптимальний розмір дерева рішень. Навести оптимальне дерево рішень.
6. Порівняти дерева рішень, які синтезовано за різними критеріями розщеплення.
7. Розрахувати матриці сплутувань найкращого дерева рішень на навчальній та тестовій вибірках та зробити за ними висновки.
8. За експериментальними даними вивести двовимірні розподіли класів рішень для кожної пари ознак та порівняти їх з правилами оптимального дерева рішень.
9. Порівняти створений класифікатор з результатами вирішення цієї задачі іншими дослідниками.

Таблиця 2 – Варіанти завдання

Варіант	Задача
1	Wilt
2	Bank Marketing
3	Car Evaluation Data Set
4	Cardiotocography
5	Activity Recognition from Single Chest-Mounted Accelerometer
6	Climate Model Simulation Crashes
7	Credit Approval

Варіант	Задача
8	Cylinder Bands
9	Ecoli
10	Glass Identification
11	Image Segmentation
12	MAGIC Gamma Telescope
13	Nursery
14	Page Blocks Classification
15	Pen-Based Recognition of Handwritten Digits
17	Spambase
18	Stalog (Shuttle)
19	Statlog (German Credit Data)
20	Statlog (Vehicle Silhouettes)
21	Steel Plates Faults
22	Banknote authentication
23	User Knowledge Modeling Data Set
24	Yeast
25	Dataset for Sensorless Drive Diagnosis

Рекомендації

Для поглибленого вивчення матеріалу рекомендуємо літературу [1–5, 8, 12, 17, 24, 25].

Під час виконання *першого завдання* бажано не лише перекласти опис задачі, який наведено в [22], але навести більше змістовної інформації. Приклад виконання цього завдання для задачі розпізнавання ірисів наведено нижче.

Розглядається задача класифікації квіток ірисів за такими чотирма ознаками:

x_1 – довжина чашолистка;

x_2 – ширина чашолистка;

x_3 – довжина пелюстки;

x_4 – ширина пелюстки.

Вибірка складається із 150 рядків. Сама вибірка доступна з http://archive.ics.uci.edu/ml/datasets/Iris .

Завдання полягає в розробці моделі, яка на основі ознак (x_1, x_2, x_3, x_4) вірно відносить ірис до одного із трьох класів:

d_1 – ірис сетоса (Iris-setosa);

d_2 – ірис веселковий (Iris-versicolor);

d_3 – ірис віржиніка (Iris-virginica).

Зображення типових ірисів кожного з класів наведено на рис. 6.

Iris-setosa Iris-versicolor Iris-virginica

Рисунок 6 – Фото ірисів з сайту http://www.signa.org

Примітка. Регіональні налаштування операційної системи, на якій створена вибірка, можуть відрізнятися від налаштувань поточної операційної системи. У разі проблем імпорту даних спробуйте розділовий знак в десяткових дробах "." замінити знаком ",". Якщо не вдається імпортувати дані з файлу MS Excel, імпортуйте їх в формат CVS за допомогою відповідного програмного додатку пакету Office, OpenOffice чи LibreOffice. Якщо при відкритті файлу в MS Excel виникає помилка, запустіть відповідний програмний додаток з пакету та "перетягніть" файл в середину або відкрийте через відповідне меню.

Під час виконання *другого завдання* слід пам'ятати, що для категоріальних змінних перевіряють на репрезентативність за розподілом їх можливих значень. У першу чергу слід забезпечити схожий розподіл класів вихідної змінної. Бажано, щоб рядки вибірки, що містять крайні значення за кожною ознакою потрапили у навчальну вибірку. Для перевірки репрезентативності вибірок за кількісними атрибутами слід розрахувати математичні сподівання та дисперсії даних за кожною

змінною. Попарно за кожною змінною ці статистичні характеристики в навчальній та тестовій вибірках мають бути приблизно однаковими.

Для розбиття даних на навчальну та тестові вибірки доцільно використовувати функції `min`, `max`, `unique`, `intersect` та `setdiff`. Для підрахунку частоти значень категоріальної змінної скористайтесь функцією `histc` в такому форматі:

```
f = histc(X, unique(X))/length(X),
```

де `f` – частота значень змінної `X`.

Для імпорту різнорідних даних, наприклад, якщо категоріальні значення задані у символьному вигляді, доцільно використати функцій `import::cvs`, `import::readdata` та `fscanf`.

Для виконання *третього завдання* доцільно використати функцію `classregtree` таким чином:

```
T = classregtree(tr_x, tr_y),
```

де `T` – дерево рішень;

`tr_x` – вхідні значення навчальної вибірки;

`tr_y` – вихідні значення навчальної вибірки.

Якщо в даних є пропускі, то у відповідні чарунки матриці `tr_x` слід занести `NaN`.

Якщо значення вектору `tr_y` задані числовими значеннями, тоді функцію `classregtree` необхідно викликати в такому форматі:

```
T=classregtree(tr_x, tr_y, 'method', 'classification').
```

Якщо деякі ознаки є категоріальними, тоді необхідно навести їх перелік. Наприклад, якщо перша і третя ознаки є категоріальними, тоді `classregtree` викликаємо таким чином:

```
T = classregtree(tr_x, tr_y, 'categorical', [1 3]).
```

Якщо відома платіжна матриця (`Cost_matrix`), тоді `classregtree` викликаємо таким чином:

```
T = classregtree(tr_x, tr_y, 'cost', Cost_matrix).
```

Для виконання *четвертого завдання* слід використовувати функції `prune` – для підрізання дерева рішень (за вершинами або за рівнями) та `eval` – для класифікації. В багатьох випадках, щоб розрахувати частоту помилок класифікації необхідно конвертувати дані у потрібний формат, наприклад, застосовуючи функції `cell2mat` та `str2num`.

Для виконання *п'ятого завдання* слід використовувати функцію `min` та `find` для знаходження оптимального дерева рішень серед класифікаторів, синтезованих під час виконання попереднього завдання. Можна використати і функцію `test`. Для візуалізації дерева рішень призначена функція `view`, яка відкриває GUI-модуль Classification tree viewer (див. рис. 4 та 5).

Для виконання *шостого завдання* функцію `classregtree` викликаємо таким чином:

```
T = classregtree(tr_x, tr_y, 'splitcriterion', value),
```

де `value` – назва методу. Запрограмовані методи: `'gdi'` – на основі індекса Джині (значення за замовченням); `'deviance'` – на основі ентропійного критерію; `'twoing'` – на основі ділення навпіл (*twoing index*). Для останнього варіанту критерій розщеплення розраховується таким чином:

$$T(v) = \frac{N_1 N_2}{4N^2}\left(\sum_{j=\overline{1,m}} abs(p_{1j} - p_{2j})\right)^2.$$

Висновки за результатами виконання *сьомого завдання* мають бути орієнтовані на потреби замовників – майбутніх користувачів розробленої вами моделі прийняття рішень. Формувати висновки слід таким чином, щоб з матриці сплутувань замовники отримали якісь корисні знання. Для розрахунку матриці сплутувань доцільно використати функцію `confusionmat`.

Для виконання *восьмого завдання* слід скористатися функцією `gplotmatrix`. Приклад результатів виконання цієї функції для задачі класифікації ірисів наведено на рис. 7. Двовимірні розподіли інформативні для візуальної перевірки класифікації за короткими гілками дерева рішень,

логічні умови яких містять 1 чи 2 вхідні атрибути. Наприклад, в дереві рішень з рис. 2 гілка для класу Versicolor складається з трьох логічних умов з атрибутами x_3 та x_4. Саме за атрибутами, що фігурують в коротких гілках дерева рішень, доцільно будувати двовимірні розподіли.

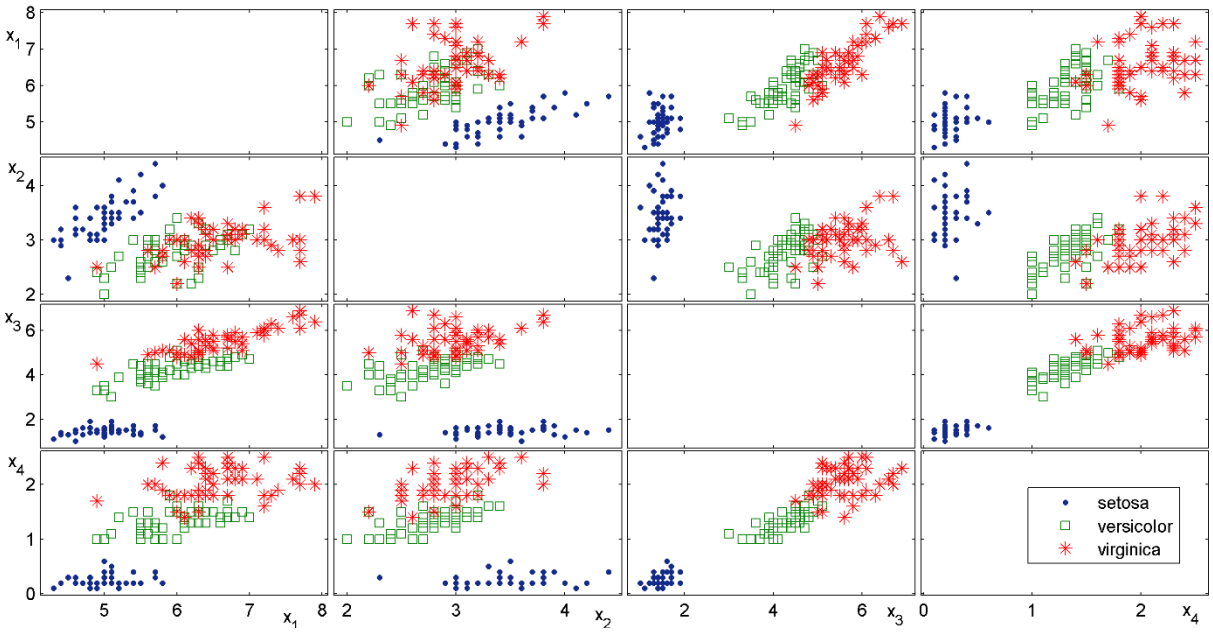


Рисунок 7 – Двофакторні розподіли класів ірисів

Виконання *дев'ятого завдання* бажано здійснити за допомогою пошукової системи наукових публікацій Google Scholar. Доцільно пошукові запити сформувати англійською мовою.

Під час виконання лабораторної роботи доцільно використовувати такі функції:

`classregtree`	–	синтез дерева рішень з даних;
`view`	–	візуалізація дерева рішень;
`eval`	–	класифікація за деревом рішень;
`prune`	–	підрізання дерева рішень;
`test`	–	знаходження оптимального дерева рішень;
`confusionmat`	–	підрахунок матриці сплутувань;
`cell2mat`	–	перетворення списку в масив;
`str2num`	–	перетворення даних з символьного формату у числові;
`max`	–	значення та порядковий номер максимального елементу масиву;

`min` – значення та порядковий номер мінімального елементу масиву;

`setdiff` – різниця множин;

`intersect` – перетин множин;

`union` – об'єднання множин;

`find` – знаходження номерів елементів масиву, які задовольняють деяку умову;

`unique` – вилучення з масиву дублюючих елементів;

`gplotmatrix` – Створення графічного зображення множини двофакторних розподілів класів.

Детальний опис функцій `classregtree`, `prune`, `view`, `eval` та `test` наведено в Довіднику ключових функцій.

Питання для самоконтролю

1. В чому принципова відмінність числової та категоріальної шкал?
2. Яка складність жадібного алгоритму синтезу дерева рішень?
3. Яка складність алгоритму класифікації за деревом рішень?
4. Який алгоритм синтезу дерев рішень реалізовано в програмі See 5 та в Statistics Toolbox програмної системи MATLAB?
5. Як проставити назви листків під час синтезу дерева рішень?
6. Як визначити платіжну матрицю для задачі класифікації позичальників під час прийняття рішень щодо кредитування?
7. Яким чином пов'язана теорема Кондорсе про присяжних з ансамблем дерев рішень?
8. Чи потрібно, щоб помилки класифікації за окремими деревами ансамблю були корельованими?
9. Чи можна в ансамбль класифікаторів включати дерева рішень, які синтезовано за різними алгоритмами?
10. Наведіть змістовні приклади помилок першого та другого родів для реальних задач класифікації?
11. Скільки типів помилок в задачі класифікації з трьома класами?
12. Які труднощі визначення платіжної матриці для реальних задач прийняття рішень?
13. Як оцінити середні витрати класифікації за деревом рішень, якщо відома вартість вимірювання кожної ознаки?

Лабораторна робота №2

Дослідження впливу кількості правил нечіткої бази знань на точність ідентифікації

Мета – побудувати криву навчання нечіткої бази знань Мамдані у формі залежності точності ідентифікації від кількості правил.

Теоретичні відомості

Труднощі ідентифікації багатьох цікавих залежностей обумовлені відсутністю достатнього обсягу точних експериментальних даних та наявністю результатів спостережень в формі нечітких знань та лінгвістичних експертних оцінок. Один із шляхів подолання цих труднощів полягає у застосуванні нечіткої ідентифікації, тобто методів побудови моделей на основі теорії нечітких множин та нечіткої логіки. Найчастіше нечітка ідентифікація проводиться коли модель залежності являє собою нечітку базу знань.

У разі моделювання залежностей за допомогою нечітких баз знань етап структурної ідентифікації полягає у визначенні вхідних та вихідних змінних моделей, виборі множин лінгвістичних значень змінних та опису залежності лінгвістичними продукційними правилами. Останні дві процедури є специфічними для нечіткої ідентифікації. В результаті структурної ідентифікації отримуємо грубу модель, яка в загальних рисах описує досліджувану залежність. На другому етапі – етапі параметричної ідентифікації підлаштовують модель, змінюючи її параметри таким чином, щоб найточніше описати залежність в експериментальних даних. У випадку нечіткої ідентифікації на цьому етапі налаштовують функції належності нечітких термів та вагові коефіцієнти правил.

Наведемо базові положення теорії нечітких множин, які необхідні для викладення матеріалу з нечіткої ідентифікації. Основою теорії нечітких

множин є ідея про те, що елементи з певної множини, які володіють деякою спільною властивістю, можуть володіти нею з різним ступенем. За такого підходу висловлювання про те, що елемент належить множині втрачає сенс, тому що необхідно вказати наскільки сильно або з яким ступенем елемент задовольняє властивостям множини.

Нечіткою множиною $\tilde{A}$ на універсальній множині U називається сукупність пар $(\mu_A(u), u)$, де $\mu_A(u)$ – ступінь належності елемента $u \in U$ до нечіткої множини $\tilde{A}$. Ступінь належності – це число з діапазону [0, 1]. Чим більший ступінь належності, тим сильніше елемент універсальної множини відповідає властивостям нечіткої множини.

Функцією належності називається така функція, яка дозволяє обчислити ступінь належності довільного елемента універсальної множини до нечіткої множини.

Якщо універсальна множина складається з кінцевого числа елементів $U = \{u_1, u_2, ..., u_k\}$, тоді нечітка множина $\tilde{A}$ записується так: $\tilde{A} = \left(\frac{\mu_A(u_1)}{u_1}, \frac{\mu_A(u_2)}{u_2}, ..., \frac{\mu_A(u_k)}{u_k} \right)$. Наприклад, нечітку множину *"чоловік середнього зросту"* можна задати такою нечіткою множиною:

$$\tilde{A} = \left(\frac{0.1}{160}, \frac{0.3}{165}, \frac{0.8}{170}, \frac{1}{175}, \frac{1}{180}, \frac{0.5}{185}, \frac{0.1}{190} \right).$$

У випадку неперервної універсальної множини U використовують запис $\tilde{A} = \int_u \mu_A(u)/u$, де знак $\int$ означає сукупність пар $\mu_A(u)$ та u.

Носієм нечіткої множини називається множина усіх елементів універсальної множини, які мають ненульовий ступінь належності. Носій нечіткої множини $\tilde{A}$ позначається $\text{supp}(\tilde{A})$.

Ядром нечіткої множини називається множина усіх елементів універсальної множини, ступінь належності яких дорівнює 1. Ядро нечіткої множини $\tilde{A}$ позначається $\text{core}(\tilde{A})$.

Доповненням нечіткої множини $\tilde{A}$ на універсумі U називається нечітка множина $\overline{A}$ з функцією належності $\mu_{\overline{A}}(u) = 1 - \mu_A(u)$ для усіх $u \in U$.

Перетином нечітких множин $\tilde{A}$ та $\tilde{B}$, які задані на U, називається нечітка множина $\tilde{C} = \tilde{A} \bigcap \tilde{B}$ з функцією належності $\mu_C(u) = \mu_A(u) \wedge \mu_B(u)$ для усіх $u \in U$, де $\wedge$ - t-норма (табл. 0).

Таблиця 3 – Найбільш уживані трикутні норми

t-норма: $a \wedge b$	s-норма: $a \vee b$	Назва
$\min(a, b)$	$\max(a, b)$	Норми Заде
ab	$a + b - ab$	Ймовірнісні норми
$\max(a + b - 1, 0)$	$\min(a + b, 1)$	Норми Лукасевича

Об'єднанням нечітких множин $\tilde{A}$ та $\tilde{B}$, які задані на U, називається нечітка множина $\tilde{D} = \tilde{A} \bigcup \tilde{B}$ з функцією належності $\mu_D(u) = \mu_A(u) \vee \mu_B(u)$ для усіх $u \in U$, де $\vee$ - s-норма (див. табл. 0).

Приклади виконання операцій доповнення, перетину та об'єднання над нечіткими множинами з використанням норм Заде наведені на рис. 8.

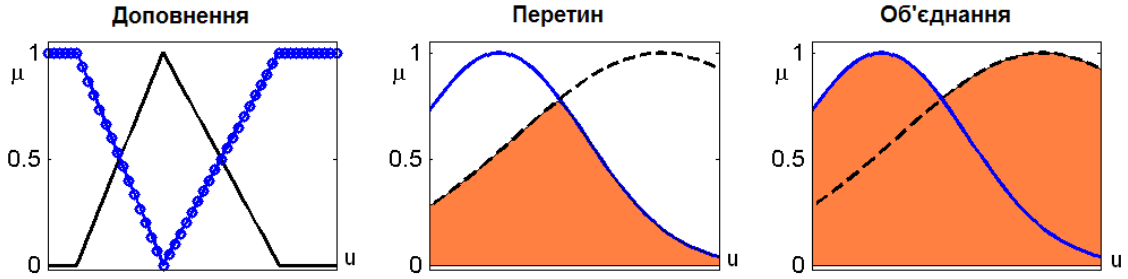


Рисунок 8 – Операції над нечіткими множинами

Лінгвістичною змінною називається змінна, значеннями якої є слова або словосполучення природної мови. Множина усіх можливих значень лінгвістичної змінної називається *терм-множиною*. Елемент терм-множини називається *термом*. Розглянемо змінну "*швидкість автомобіля*", яка оцінюється за шкалою "*низька*", "*середня*", "*висока*" та "*дуже висока*". В цьому прикладі лінгвістичною змінною є "*швидкість автомобіля*", термами є лінгвістичні оцінки "*низька*", "*середня*", "*висока*" та "*дуже висока*", які і складають терм-множину.

В теорії нечітких множин терм задається функцією належності. При побудові функцій належностей за експертними думками найбільше

поширення отримали методи на основі парних порівнянь та статистичного оброблення експертних оцінок.

За першим методом кожен експерт заповнює анкету, в якій вказує на наявність або відсутність у елемента u_i $(i=\overline{1,k})$ властивостей нечіткої множини $\tilde{l}_j$ $(j=\overline{1,m})$. Сукупність елементів $u_1, u_2, ..., u_k$ утворює універсум для нечітких множин $\tilde{l}_1$, $\tilde{l}_2, ..., \tilde{l}_m$. Анкета має вигляд такої таблиці:

	u_1	u_2	…	u_k
$\tilde{l}_1$				
$\tilde{l}_2$				
…				
$\tilde{l}_m$				

Введемо такі позначення: V – кількість експертів; g_{ji}^{v} – думка v-го експерта про наявність у елемента u_i властивостей нечіткої множини $\tilde{l}_j$, $v=\overline{1,V}$, $i=\overline{1,k}$, $j=\overline{1,m}$. Якщо $g_{ji}^{v}=1$, тоді експерт вважає, що властивості наявні, а якщо $g_{ji}^{v}=0$, тоді – відсутні. За результатами анкетування ступені належності розраховують так:

$$\mu_{l_j}(u_i)=\frac{1}{V}\sum_{v=\overline{1,V}} g_{ji}^{v},\ \ i=\overline{1,k},\ \ j=\overline{1,m}.$$

За другим методом експерт оцінює перевагу елемента u_i над u_j по відношенню до властивостей нечіткої множини $\tilde{l}$ $(i,j=\overline{1,k})$. Парні порівняння зручно задавати такою матрицею:

$$\mathbf{A}=\begin{array}{c} \\ u_1 \\ u_2 \\ \dots \\ u_k \end{array}\begin{array}{c} \begin{array}{cccc} u_1 & u_2 & \dots & u_k \end{array} \\ \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1k} \\ a_{21} & a_{22} & \dots & a_{2k} \\ \dots & & & \\ a_{k1} & a_{k2} & \dots & a_{kk} \end{bmatrix} \end{array},$$

де a_{ij} – рівень переваги u_i над u_j $(i,j=\overline{1,k})$ за шкалою Сааті.

Шкала Сааті є такою:

1 – перевага *відсутня*;

2 – *ледь слабка* перевага;

3 – *слабка* перевага;

4 – більш ніж слабка перевага;

5 – *помірна* перевага;

6 – *майже сильна* перевага;

7 – *сильна* перевага;

8 – *майже абсолютна* перевага;

9 – *абсолютна* перевага.

Матриця **A** є діагональною ($a_{ii} = 1$) та зворотно симетричною ($a_{ij} = 1/a_{ji},\ i, j = \overline{1, k}$).

Ступеням належності нечіткої множини відповідають нормалізовані координати власного вектору $W = (w_1, w_2, ..., w_k)^{\mathrm{T}}$ матриці **A**:

$$\mu_l(u_i) = \frac{w_i}{\max(w_1, w_2, ..., w_k)}, \quad i = \overline{1, k}.$$

Власний вектор знаходять з такої системи рівнянь:

$$\begin{cases} \mathbf{A} \cdot W = \lambda_{\max} \cdot W \\ w_1 + w_2 + ... + w_v = 1, \end{cases}$$

де $\lambda_{\max}$ – найбільше власне значення матриці **A**.

В результаті застосування вищеописаних методів отримуємо дискретну нечітку множину, тобто кінцевий список дійсних чисел та ступенів належності:

$$\tilde{l} = \left(\frac{\mu_l(u_1)}{u_1}, \frac{\mu_l(u_2)}{u_2}, ..., \frac{\mu_l(u_k)}{u_k} \right),$$

де u_i – елемент універсальної множини, $i = \overline{1, k}$;

$\mu_l(u_i)$ – ступінь належності елемента u_i до нечіткої множини $\tilde{l}$.

Для переходу на неперервну універсальну множину дійсних чисел апроксимуємо знайдені ступені належності. Це може бути або кусково-

лінійна інтерполяція, або апроксимація деякими параметричними функціями належності. В останньому випадку за критерій точності апроксимації оберемо середню квадратичну нев'язку:

$$RMSE = \sqrt{\frac{1}{k}\sum_{i=\overline{1,k}}\left(\mu_l(u_i) - \mathrm{mf}(P, u_i)\right)^2},$$

де $\mathrm{mf}(P, u_i)$ – значення функції належності mf з параметрами P для аргументу u_i.

Параметричні функції належності зазвичай мають 2, 3 або 4 параметри. Існує багато типів параметричних функцій належностей, найбільш розповсюдженими серед яких є трикутна, трапецієва, гаусова та дзвонова. Аналітичні вирази цих функцій належностей та їх графічне зображення зведені в табл. 0.

Нечітка логіка – це різновид багатозначної логіки, в якій значення істинності задаються лінгвістичними змінними або такими термами лінгвістичної змінної "істинність" як: "дуже істинно", "майже істинно", "трохи хибно" тощо. Ці лінгвістичні значення істинності описуються нечіткими множинами. Правила виконання нечітких логічних операцій отримують з бульових логічних операцій за допомогою принципу нечіткого узагальнення.

Позначимо нечіткі логічні змінні через $\tilde{A}$ і $\tilde{B}$, а функції належності, що задають істинності значення цих змінних через $\mu_{\tilde{A}}(u)$ і $\mu_{\tilde{B}}(u)$, $u \in [0,1]$. *Нечіткі логічні операції* ТА ($\wedge$), АБО ($\vee$), НІ ($\bar{}$) та імплікація ($\Rightarrow$) виконуються за такими правилами:

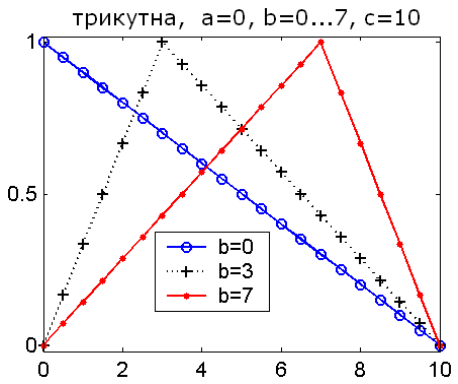
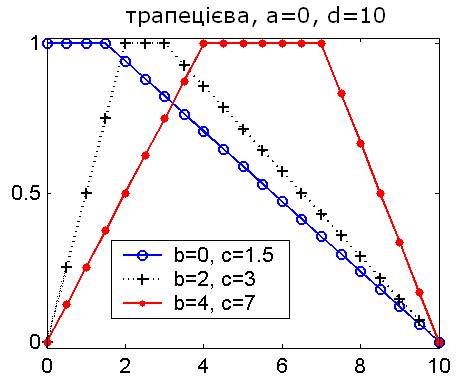
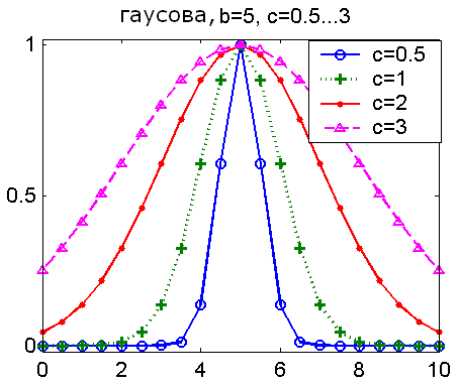
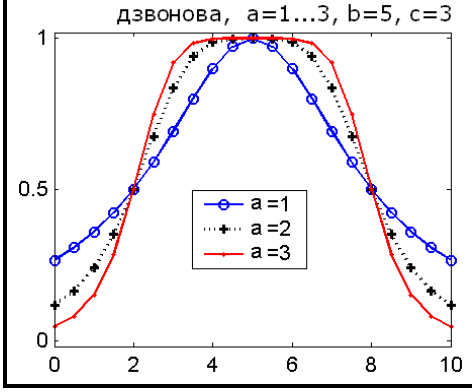
$$\mu_{A \wedge B}(u) = \min\left(\mu_A(u), \mu_B(u)\right);$$

$$\mu_{A \vee B}(u) = \max\left(\mu_A(u), \mu_B(u)\right);$$

$$\mu_{\bar{A}}(u) = 1 - \mu_A(u);$$

$$\mu_{A \Rightarrow B}(u) = \max\left(1 - \mu_A(u), \mu_B(u)\right).$$

Таблиця 4 – Популярні параметричні функцій належності

Функція належності	Аналітичний вираз
	$\mu(x)=\begin{cases}0, & \text{якщо } y<a\\ \dfrac{x-a}{b-a}, & \text{якщо } a\le x<b\\ \dfrac{c-x}{c-b}, & \text{якщо } b\le x\le c\\ 0, & \text{якщо } x>c\end{cases},$ де b – координата максимуму; (a, c) – носій нечіткої множини $\tilde{x}$.
	$\mu(x)=\begin{cases}0, & \text{якщо } x<a\\ \dfrac{x-a}{b-a}, & \text{якщо } a\le x<b\\ 1, & \text{якщо } b\le x<c\\ \dfrac{d-x}{d-c}, & \text{якщо } c\le x\le d\\ 0, & \text{якщо } x>d\end{cases},$ де $[b, c]$ – ядро нечіткої множини $\tilde{x}$; (a, d) – носій нечіткої множини $\tilde{x}$.
	$\mu(x)=\exp\left(-\dfrac{(x-b)^2}{2c^2}\right),$ де b – координата максимуму; c – коефіцієнт концентрації.
	$\mu(x)=\dfrac{1}{1+\left\lvert\dfrac{x-b}{c}\right\rvert^{2a}},$ де a – коефіцієнт крутизни; b – координата максимуму; c – коефіцієнт концентрації.

Нечіткою базою знань називається сукупність нечітких правил <Якщо – тоді>, які описують певну предметну область. Якщо-частина правила називається *антецедентом* або *посилкою*, а тоді-частина правила – *консеквентом* або *висновком*. Найуживанішими є нечіткі синглтонна база знань, класифікаційна база знань, база знань Мамдані, база знань Сугено, база знань Ларсена та база знань Цукамото. В лабораторній роботі досліджується база знань Мамдані.

В базі знань Мамдані антецеденти і консеквенти задано нечіткими множинами. Цю базу знань можна трактувати як розбиття факторного простору на зони з нечіткими межами, в кожній з яких функція відклику приймає нечітке значення. Кількість нечітких зон дорівнює числу правил. Нечітку базу знань Мамдані, що описує залежність $y = f(x_1, x_2, \ldots, x_n)$, представимо таким чином:

$$\begin{array}{l} \text{Якщо } \left(x_1 = \tilde{a}_{1j} \text{ та } x_2 = \tilde{a}_{2j} \text{ та ... та } x_n = \tilde{a}_{nj} \text{ з вагою } w_j\right), \\ \text{тоді } y = \tilde{d}_j, \quad j = \overline{1, N}, \end{array} \tag{6}$$

де $\tilde{d}_j$ – нечітке значення, яке обирається з терм-множини $\{\tilde{l}_1, \tilde{l}_2, \ldots, \tilde{l}_m\}$, кожен елемент якої представлено нечіткою множиною $\tilde{l}_s = \int\limits_{y \in [\underline{y}, \overline{y}]} \mu_{l_s}(y)/y$,

$s = \overline{1, m}$;

$w_j \in [0,1]$ – ваговий коефіцієнт, який віддзеркалює впевненість експерта в адекватності *j*-го правила;

N - кількість правил в базі знань.

Як приклад наведемо фрагмент нечіткої бази знань Мамдані про залежність рівня безпомилковості (y) людини-оператора від тривалості роботи (x_1) та напруженості роботи (x_2):

якщо $x_1 = Мала$, тоді $y = Низький$;

якщо $x_1 = Середня$ та $x_2 = Середня$, тоді $y = Високий$;

якщо $x_1 = Велика$ та $x_2 = Низька$, тоді $y = Низький$;

якщо $x_1 = Велика$ та $x_2 = Велика$, тоді $y = Низький$;

якщо $x_1 = Велика$ та $x_2 = Середня$, тоді $y = Середня$.

Нечітку базу знань (6) зручно подавати у формі табл. 0. Для адекватного опису залежності $y = f(x_1, x_2, \ldots, x_n)$ в деякій області факторного простору не завжди необхідні значення усіх вхідних змінних. Для таких випадків правила бази знань можна представити в неповному форматі, вилучивши входи, які не впливають на вихідне значення. Для врахування такої особливості в нечіткій базі знань позначимо символом "–" змінні, що можуть приймати довільні значення без порушення істинності відповідного правила. Інколи для цього використовують нечітку множину "*Don't care*", функція належності якої дорівнює 1 на усьому діапазоні значень відповідної вхідної змінної.

Таблиця 5 – Нечітка база знань Мамдані

Якщо				Тоді
x_1	x_2	…	x_n	y
$\tilde{a}_{11}$	$\tilde{a}_{21}$	…	$\tilde{a}_{n1}$	d_1
$\tilde{a}_{12}$	$\tilde{a}_{22}$	…	$\tilde{a}_{n2}$	d_2
...				
$\tilde{a}_{1N}$	$\tilde{a}_{2N}$	…	$\tilde{a}_{nN}$	d_N

Логічне виведення за нечіткою базою знань (6) здійснюють за схемою з рис. 9. Спочатку для поточного вхідного вектора $X^* = \left(x_1^*, x_2^*, \ldots, x_n^*\right)$ розраховують ступінь виконання антецедента *j*-го правила:

$$\mu_j(X^*) = w_j \cdot \left(\mu_j(x_1^*) \wedge \mu_j(x_2^*) \wedge \ldots \wedge \mu_j(x_n^*)\right), \quad j = \overline{1, N},$$

де $\wedge$ – t-норма, яку в алгоритмі Мамдані зазвичай реалізують операцією мінімуму.

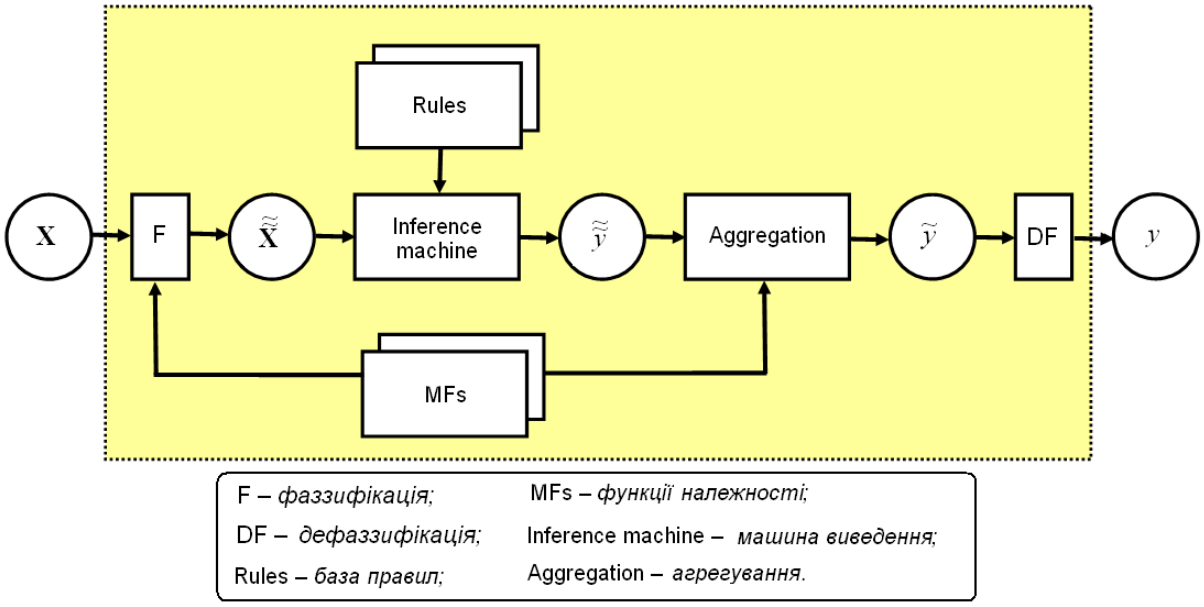


Рисунок 9 – Логічне виведення за нечіткою базою знань Мамдані

Результат логічного виведення можна записати у формі такої бінечіткої множини:

$$\tilde{\tilde{y}}^* = \left(\frac{\mu_{d_1}(X^*)}{\tilde{d}_1}, \frac{\mu_{d_2}(X^*)}{\tilde{d}_2}, ..., \frac{\mu_{d_N}(X^*)}{\tilde{d}_N} \right),$$

особливістю якої є те, що елементами її носія є нечіткі множини $\tilde{d}_1, \tilde{d}_2, ..., \tilde{d}_N$. Для перетворення $\tilde{\tilde{y}}^*$ в звичайну нечітку множину виконаємо такі дії. Спочатку представимо результат виведення за j-им правилом бази знань у формі такої нечіткої множина:

$$\tilde{d}_j^* = \text{imp}\left(\tilde{d}_j,\ \mu_j(X^*)\right),\quad j = \overline{1, N}, \tag{7}$$

де imp позначає імплікацію, яку реалізують операцією мінімуму.

Геометричною інтерпретацією імплікації є зрізання графіка функції належності $\mu_{d_j}(y)$ по рівню $\mu_j(X^*)$, що математично запишемо так:

$$\tilde{d}_j^* = \int_{y \in [\underline{y}, \bar{y}]} \min\left(\mu_j(X^*),\ \mu_{d_j}(y)\right) / y\,.$$

Результат виведення за усіма правилами знаходять агрегуванням нечітких множин (7):

$$\tilde{y}^* = \text{agg}\left(\tilde{d}_1^*, \tilde{d}_2^*, ..., \tilde{d}_N^*\right),$$

де agg – агрегування нечітких множин, яке реалізують операцією максимуму. Ілюстрацією цієї формули є рис. 10, де здійснюється агрегування трьох нечітких множин.

Чітке значення виходу y^*, яке відповідає вхідному вектору X^*, визначається через дефаззифікацію нечіткої множини $\tilde{y}^*$. Популярні методи дефаззифікації зведено в табл. 0.

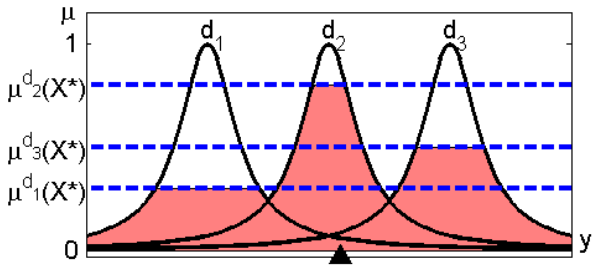


Рисунок 10 – Імплікація, агрегування та дефаззифікація в алгоритмі Мамдані

Таблиця 6 – Дефаззифікація різними методами нечіткої множини

$$\tilde{A} = \int_{u \in [\underline{u}, \bar{u}]} \mu_A(u)/u$$

Назва методу дефаззифікації	Розрахункова формула
Центр тяжіння	$\dfrac{\int_{\underline{u}}^{\bar{u}} u \cdot \mu_A(u)\, du}{\int_{\underline{u}}^{\bar{u}} \mu_A(u)\, du}$
Медіана	знайти таке число a, щоб: $\int_{\underline{u}}^{a} \mu_A(u)\, du = \int_{a}^{\bar{u}} \mu_A(u)\, du$
Центр максимумів	$\dfrac{\int_G u du}{\int_G du}$, де $G = \arg \sup_{u \in \text{supp}(\tilde{A})} (\mu_A(u))$

Лабораторна робота стосується структурної ідентифікації за допомогою нечіткої бази знань. Завдання полягає у генеруванні множини нечітких правил, які описують особливості досліджуваної залежності. Далі з цих правил-кандидатів необхідно сформувати нечіткі бази знань різної розмір-ності та дослідити як впливає кількість правил на точність ідентифікації.

В ідеальному випадку нечітка база знань має бути і компактною, і адекватною. Досягти цього в реальних задачах неможливо, тому на практиці намагаються обрати базу знань з коректним балансом між цими суперечливими критеріями. Необхідною умовою такого балансу є потрапляння бази знань на Парето-фронт у координатах "складність моделі – точність моделі".

Точність нечіткої моделі оцінимо за допомогою середньої квадратичної нев'язки ($RMSE$). Розглянемо залежність з n входами $X=(x_1, x_2, ..., x_n)$ та одним виходом y. Вважається відомою вибірка експериментальних даних про цю залежність:

$$(X_r, y_r),\ \ r=\overline{1, M}, \tag{8}$$

де M – довжина вибірки;

$X_r=(x_{r1}, x_{r2}, ..., x_{rn})$.

У вибірці даних вхідні та вихідна змінні приймають числові значення. Позначимо нечітку модель цієї залежності через $y=F(X)$. Тоді середня квадратична нев'язка між експериментальними та модельними результатами розраховується таким чином:

$$RMSE=\sqrt{\frac{1}{M}\sum_{r=\overline{1, M}}(y_r-F(X_r))^2}\ . \tag{9}$$

Для оцінювання складності нечіткої моделі використовують переважно такі показники:

N – кількість правил в базі знань;

N_{r1} – число правил в базі знань, в антецеденті яких є лише 1 змінна, тобто кількість антецедентів довжиною в 1 елемент;

N_{r2} – число правил в базі знань, в антецеденті яких є лише 2 змінні тобто кількість антецедентів довжиною в 2 елементи;

N_{r3} – число правил в базі знань в антецеденті, яких є лише 3 змінні, тобто кількість антецедентів довжиною в 3 елементи;

N_{vr} – сумарна довжина антецедентів усіх правил бази знань;

N_{x_i} – потужність терм-множини вхідної змінної x_i, $i=\overline{1,n}$;

$N_x^{total} = \sum_{i=\overline{1,n}} N_{x_i}$ – сумарна кількість термів вхідних змінних;

$RF = \dfrac{N}{N_{\max}}$ – рівень наповненості бази знань правилами, де $N_{\max} = \prod_{i=\overline{1,n}} N_{x_i}$ – максимально можлива кількість правил;

$AF = \dfrac{N_{vr}}{n \cdot N_{\max}}$ – рівень наповненості антецедентів правил бази знань.

Відбір правил нечіткої бази знань можна звести до оптимізаційної задачі про рюкзак. Правилу бази знань відповідає предмет, який може потрапити до рюкзака, точності бази знань – корисність рюкзака, а кількості правил – сумарна об'єм обраних предметів. Відмінність між задачами полягає в різних типах функції корисності, яка є лінійною в задачі про рюкзак та нелінійною в задачі відбору правил бази знань. Аналогічно до класичних постановок задачі про рюкзак сформовано й задачі відбору правил. Задача відбору правил, як і задача про рюкзак, є NP-повною. Відповідно алгоритм точного розв'язання цієї задачі матиме експоненціальну обчислювальну складність, і тому буде прийнятним лише за невеликої кількості правил-кандидатів.

Розглянемо приклад експериментального дослідження залежності точності ідентифікації *RMSE* від кількості правил *N* нечіткої бази знань Мамдані. Дослідження проведемо для такої еталонної залежності (рис. 11):

$$y = x_1\sqrt{x_2}, \quad x_1 \in [2;22],\ x_2 \in [2;14]. \tag{10}$$

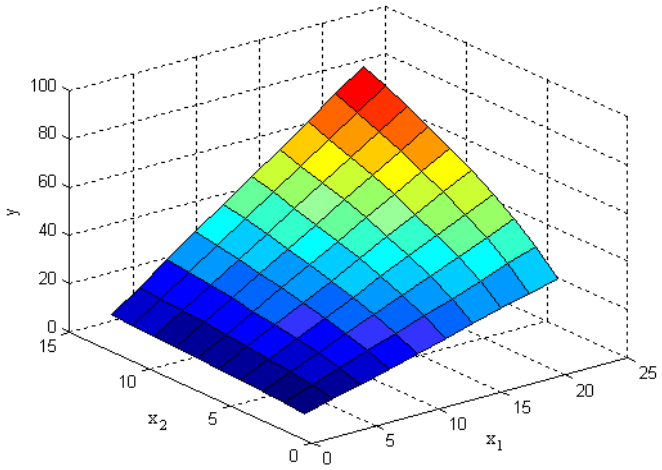


Рисунок 11 – Еталонна залежність

За фіксованого нечіткого розбиття вхідних та вихідної змінних можна згенерувати кілька нечітких баз знань з одним і тим самим числом правил N. Задачу дослідження поставимо як знаходження залежності точності $RMSE$ від обсягу N бази знань для найкращого, найгіршого та середнього випадків.

Для кожної нечіткого розбиття експерименти проведемо за такою схемою:

- згенерувати тестову вибірку з 100 точок;
- згенерувати повний список з $N_{\max}$ адекватних нечітких правил;
- синтезувати усі можливі нечіткі бази з N правил, $N = \overline{1, N_{\max}}$;
- для кожної нечіткої бази знань розрахувати нев'язку $RMSE$ на тестовій вибірці;
- для кожної множини нечітких баз знань одного розміру знайти мінімальну $RMSE_{min}$, максимальну $RMSE_{max}$ та середню $RMSE_{mean}$ нев'язки;
- побудувати графіки залежностей $RMSE_{min}$, $RMSE_{mean}$ та $RMSE_{max}$ від N.

Нечітке розбиття здійснимо за допомогою гаусових функцій належності з рівномірним розподілом ядер нечітких множин по діапазону вхідних змінних. Коефіцієнт концентрації функцій належності прийнято рівним $c = \Delta core / 2.4$, де $\Delta core$ – відстань між ядрами сусідніх термів. За такого розподілу висота перетину нечітких множин сусідніх термів дорівнює 0.5.

Для оцінювання вхідних змінних застосуємо 2, 3 та 4 термів, тобто експерименти проведено для таких 9-ти нечітких розбиттів вхідних

змінних: 2x2, 2x3, 2x4, 3x2, 3x3, 3x4, 4x2, 4x3 та 4x4. Максимальна кількість адекватних нечітких правил ($N_{\max}$) склала 4, 6, 8, 6, 9, 12, 8, 12 та 16. На протязі одного обчислювального експерименту перевірялось від $2^{2\cdot 2}-1=15$ до $2^{4\cdot 4}-1=65535$ нечітких баз знань. Відповідно, здійснено від 1500 до 6553500 нечітких логічних виведень.

Результати експериментів (рис. 12) показали, що усереднена нев'язка $RMSE_{mean}$ спадає зі збільшенням кількості нечітких правил і досягає мінімуму за повної бази знань. Графік нев'язки для найкращих комбінацій правил ($RMSE_{min}$) відповідає Парето-фронту. Кожна база знань з Парето-фронту є найкращою моделлю з заданим рівнем складності N. На Парето-фронті добре простежується плато насичення, коли додавання нових правил майже не змінює адекватність нечіткої моделі.

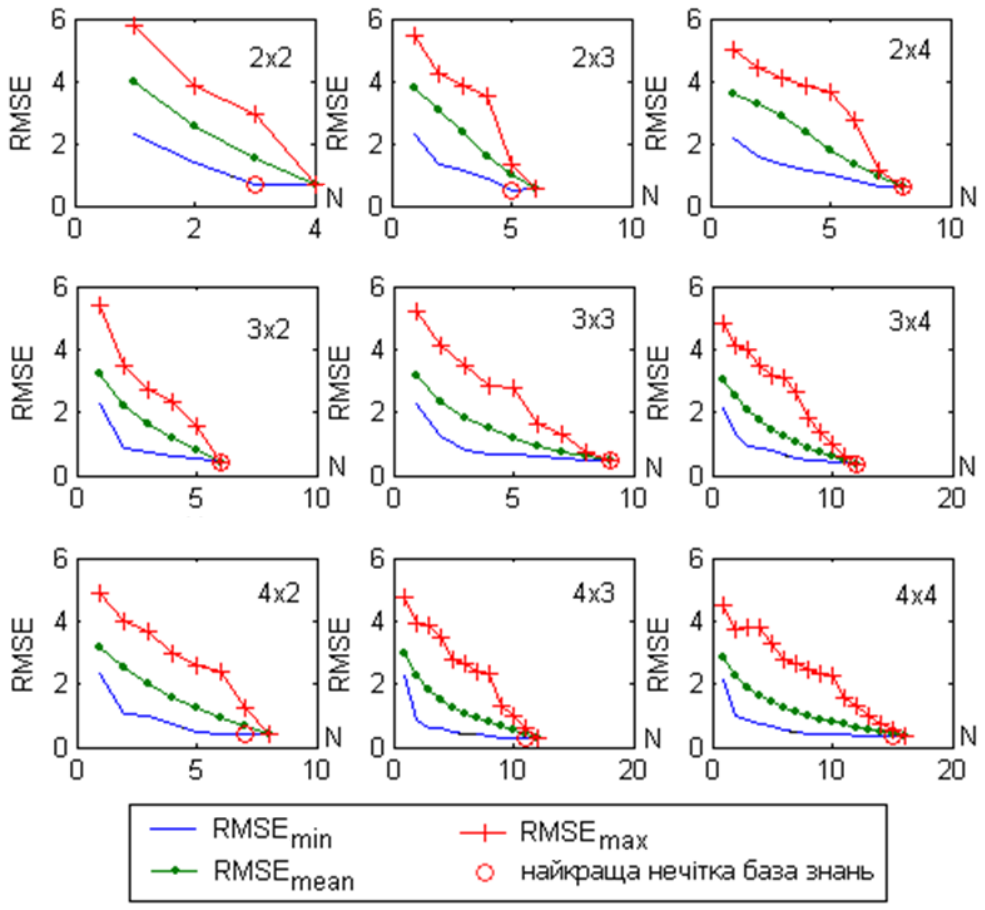


Рисунок 12 – Криві навчання баз знань Мамдані для залежності (10)

Завдання на лабораторну роботу

В лабораторній роботі проектується нечітка база знань Мамдані, яка моделює еталонну залежність $y = f(x_1, x_2)$ з табл. 0. В цій таблиці також вказані потужності терм-множин вхідних та вихідної змінних нечіткої бази знань Мамдані. За графічним зображенням залежності $y = f(x_1, x_2)$ слід згенерувати правила нечіткої бази знань. Далі потрібно експериментально встановити залежність точності ідентифікації від кількості правил нечіткої бази знань.

Таблиця 7 – Варіанти завдань

Варі-ант	Аналітична залежність	Кількість термів			Діапазон	
		x_1	x_2	y	x_1	x_2
1	$y = x_1\sqrt{x_2}$	5	4	4	[2, 22]	[3, 16]
2	$y = \frac{{x_1}^2 + {x_2}^3}{x_1 + x_2}$	4	3	4	[0, 20]	[6, 10]
3	$y = \frac{x_1 + x_1x_2 - \sqrt{x_2}}{x_1 + x_2}$	4	4	5	[1, 5]	[0.5, 2]
4	$y = \frac{x_1 + x_2}{\sqrt{x_1} + {x_2}^2 - 1}$	4	5	4	[0.6, 2]	[1, 3]
5	$y = \frac{{x_1}^3 \sin(0.2x_2)}{x_1 + {x_2}^2}$	3	4	5	[100, 150]	[50, 125]
6	$y = (1 + \sin(x_1)^2)^{x_2}$	5	4	4	[0, 3]	[0.5, 2]
7	$y = \frac{\sin x_1}{x_2}$	4	3	5	[0, 5]	[1, 10]
8	$y = \frac{{x_1}^2 - {x_2}^3}{\sqrt{2x_1} + x_2}$	4	4	5	[0, 4]	[2, 6]
9	$y = \frac{x_1 - \sqrt{x_2}}{\sqrt{x_1} + x_2}$	4	5	4	[0, 0.5]	[0.05, 0.5]
10	$y = \frac{x_1 - {x_2}^3}{x_1x_2}$	3	4	5	[10, 25]	[1, 7]
11	$y = (x_1 - 4) \cdot (x_2 - 5)^3$	5	4	4	[20,80]	[3, 12]
12	$y = 7\sqrt{x_1} - 6x_1x_2$	4	3	5	[30, 50]	[50, 100]

Варі-ант	Аналітична залежність	Кількість термів			Діапазон	
		x_1	x_2	y	x_1	x_2
13	$y = x_1^4 \sin(x_2) - \sqrt{x_2}$	4	4	5	[3, 6]	[5, 10]
14	$y = x_1^2 - x_2^3 \operatorname{tg}(0.1x_1)$	4	5	4	[-2, 3]	[1, 5]
15	$y = \frac{x_1 + x_2^3}{2\sqrt{x_1 x_2}}$	3	4	5	[2, 7]	[5, 22]
16	$y = 0.01 x_1^2 tg(x_2)$	5	4	4	[-20, 25]	[-1, 1.3]
17	$y = (\sqrt{x_1} + x_2)\sin(x_1 + x_2)$	4	3	5	[1, 4]	[0,5, 2.5]
18	$y = \frac{x_1^2 + \sqrt{x_2}}{e^{x_1 x_2}}$	4	4	5	[-4, -3]	[0, 1]
19	$y = \sin(x_1)\cos(2x_2)$	4	5	4	[1, 2]	[1, 2]
20	$y = \frac{\sqrt{x_1^2 + x_2^2}}{x_1 + x_2}$	3	4	5	[2, 4]	[3, 5]
21	$y = \frac{x_1^2 + \sin(0.1x_2)}{\cos x_1 + x_2}$	5	4	4	[1, 5]	[3, 7]
22	$y = \frac{x_1^2 + x_2}{0.1e^{x_1} + x_2}$	4	3	5	[-5, 5]	[3, 7]
23	$y = (x_1 x_2)\sqrt{x_1^2 + x_2^2}$	4	4	5	[5, 20]	[-10, 10]
24	$y = (x_1 - x_2)(x_1^2 + x_1 x_2 + x_2^2)$	4	5	4	[0.1, 2]	[0, 1.5]

Лабораторна робота полягає у виконанні таких завдань.

1. Побудувати тривимірний графік заданої аналітичної залежності.

2. Спостерігаючи за тривимірним графіком аналітичної залежності сформувати повну базу знань нечітких правил типу Мамдані.

3. Згенерувати тестову вибірку обсягом 100 точок.

4. Побудувати 3 криві навчання у формі залежності нев'язки *RMSE* від кількості правил N нечіткої бази знань. Експерименти провести для $N = \overline{1, N_{\max}}$. Порядок додавання правил в базу знань має бути унікальними для кожної кривої навчання. Узагальнити результати експериментів, провівши криві навчання $RMSE_{min}$, $RMSE_{max}$ та $RMSE_{mean}$.

5. Побудувати криву навчання у формі залежності *RMSE* від N, використовуючи жадібний алгоритм відбору правил нечіткої бази знань.

Експерименти провести для $N = \overline{1, N_{\max}}$. За жадібним алгоритмом на кожній ітерації в базу знань додається правило, яке забезпечує найбільше зменшення нев'язки $RMSE$.

6. Синтезувати усі можливі варіанти бази знань з N правил, $N = \overline{1, N_{\max}}$. Провести узагальнення результатів експериментів, провівши криві навчання $RMSE_{min}$, $RMSE_{max}$ та $RMSE_{mean}$.

Лабораторна робота має 4 рівні складності:

1 рівень – завдання 1, 2, 3 та 4;

2 рівень – завдання 1, 2, 3 та 5;

3 рівень – завдання 1, 2, 3 та 6;

4 рівень – завдання 1, 2, 3, 5 та 6.

Рекомендації

Для поглибленого вивчення матеріалу рекомендуємо літературу [5, 6, 8, 10, 11, 14–19, 21]. Доцільно ознайомитись з описом GUI-модулів пакету Fuzzy Logic Toolbox з [21].

Під час виконання *першого завдання* бажано скористатися функціями побудови тривимірних поверхонь `surf` або `mesh`. Нижче наведено приклад програми побудови графіка функції $y = x_1^2 \cdot \sin(x_2 - 1)$ в області $x_1 \in [-7, 3]$ та $x_2 \in [-4.4, 1.7]$. Результати роботи програми показано на рис. 13.

```
%Побудова графіка функції y=x1^2*sin(x2-1)
%в області x1є[-7,3] та x2є[-4.4,1.7].
n=15;   . %кількість точок дискретизації.
%Розбиваємо інтервал на n рівних відрізків:
range1 = linspace(-7, 3, n);
range2 = linspace(-4.4, 1.7, n);
f = @(x1, x2) (x1.^2).*sin(x2 -1);   %функція
%Створення декартової сітки nxn, по якій побудуємо графік:
[x1, x2] = meshgrid(range1, range2);
y=f(x1,x2);
surf(x1,x2,y);
xlabel('x_1');
ylabel('x_2');
zlabel('y', 'rotation', 0)
xlim(minmax(range1));
ylim(minmax(range2));
colormap('autumn')
```

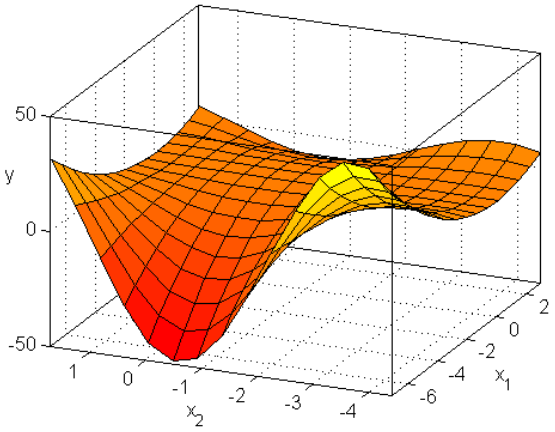


Рисунок 13 – Графік функції $y = x_1^2 \cdot \sin(x_2 - 1)$

Друге завдання полягає у проектуванні системи нечіткого виведення. Виконання цього завдання складається з 17-ти кроків.

Крок 1. Завантажити fis-редактор - редактор системи нечіткого виведення, надрукувавши в командному рядку слова `fuzzy`. Після чого з'явиться вікно з рис. 14.

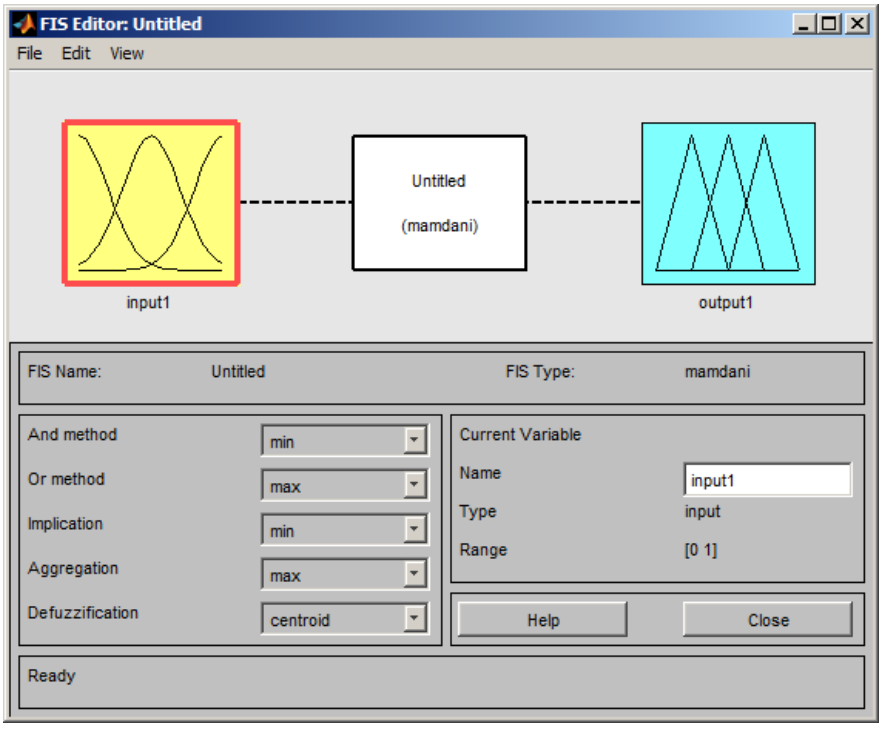


Рисунок 14 – Головне вікно fis-редактору

Крок 2. Додати другу вхідну змінну обравши пункт Add Input в меню Edit.

Крок 3. Перейменувати першу вхідну змінну. Для цього натиснути лівою кнопкою миші на блоці Input1, ввести нове позначення `x_1` в поле редагування назви поточної змінної і натиснути <Enter>.

Крок 4. Перейменувати другу вхідну змінну. Для цього натиснути лівою кнопкою миші на блоці input2, ввести нове позначення `x_2` в поле редагування назви поточної змінної і натиснути <Enter>.

Крок 5. Перейменувати вихідну змінну. Для цього натиснути лівою кнопкою миші на блоці output1, ввести нове позначення `y` в поле редагування назви поточної змінної і натиснути <Enter>.

Крок 6. Задати назву системі. Для цього в меню File вибрати в підменю Export пункт To File... і ввести ім'я файлу, наприклад, `first`.

Крок 7. Перейти в редактор функцій належності, подвійно натиснувши лівою кнопкою миші на блоці x_1.

Крок 8. Задати діапазон зміни змінної x_1. Для цього надрукувати `-7 3` в поле Range і натиснути <Enter>.

Крок 9. Задати функції належності змінної x_1. Для лінгвістичної оцінки цієї змінної використаємо 3 терми з трикутними функціями належності. Якщо в вікні немає ще функцій належності, тоді в меню Edit слід вибрати команду Add MFs.... В результаті з'явиться діалогове вікно вибору типу і кількості функцій належності. За замовченням - це 3 терми з трикутною функцією належності. Тому просто натискаємо <Enter>.

Крок 10. Задати терми змінної x_1. Для цього натискаємо лівою кнопкою миші на графіку першої функції належності (рис. 15). Потім вводимо найменування терму, наприклад, `Низький`, в полі Name і натискаємо <Enter>. Потім натискаємо лівою кнопкою миші на графіку другої функції належності і вводимо найменування терму, наприклад, `Середній`, в полі Name і натискаємо <Enter>. Ще раз натискаємо лівою кнопкою миші по графіку третьої функції належності, вводимо найменування терму, наприклад, `Високий`, в полі Name і натискаємо <Enter>. В результаті отримуємо графічне вікно, яке зображено на рис. 15.

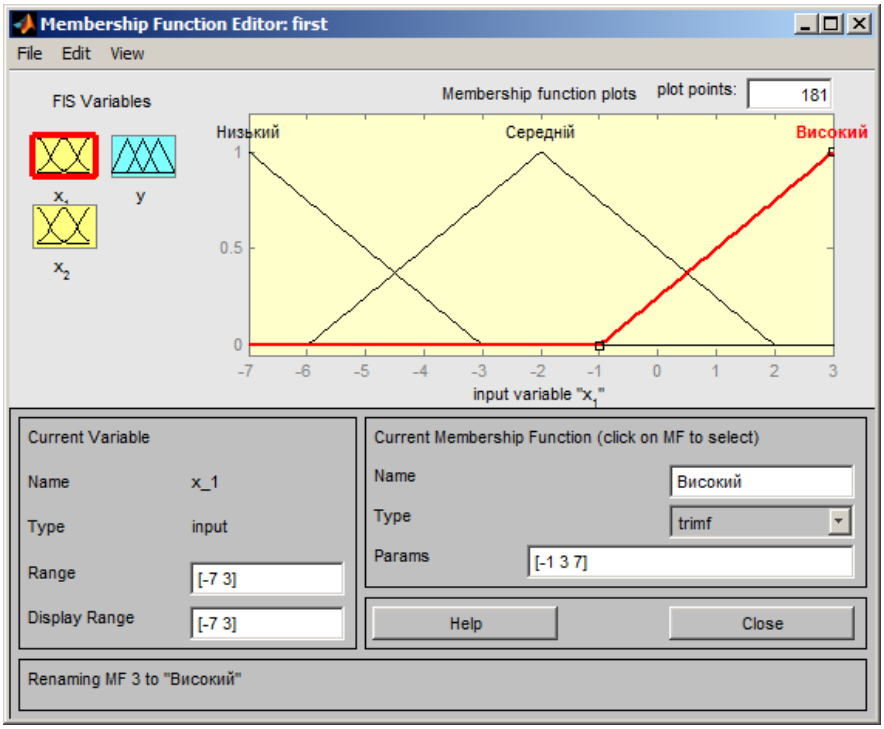


Рисунок 15 – Функції належності змінної x_1

Крок 11. Задамо функції належності змінної x_2. Для лінгвістичної оцінки цієї змінної будемо використовувати 3 терми з трикутними функціями належності. Якщо в вікні немає ще функцій належності, тоді в меню Edit слід вибрати команду Add MFs.... В результаті з'явиться діалогове вікно вибору типу і кількості функцій належності. За замовченням - це 3 терми з трикутною функцією належності. Тому просто натискаємо <Enter>.

Крок 12. За аналогією з кроком 10 задамо наступні найменування термів змінної x_2: "Низький", "Середній" та "Високий". Змінимо параметри термів в полі Params, так щоб на носій терма "Середній" припадав весь вхідний інтервал [-4.4, 1.7], а на носії крайніх термів – по половині цього інтервалу. В результаті отримуємо графічне вікно з рис. 16.

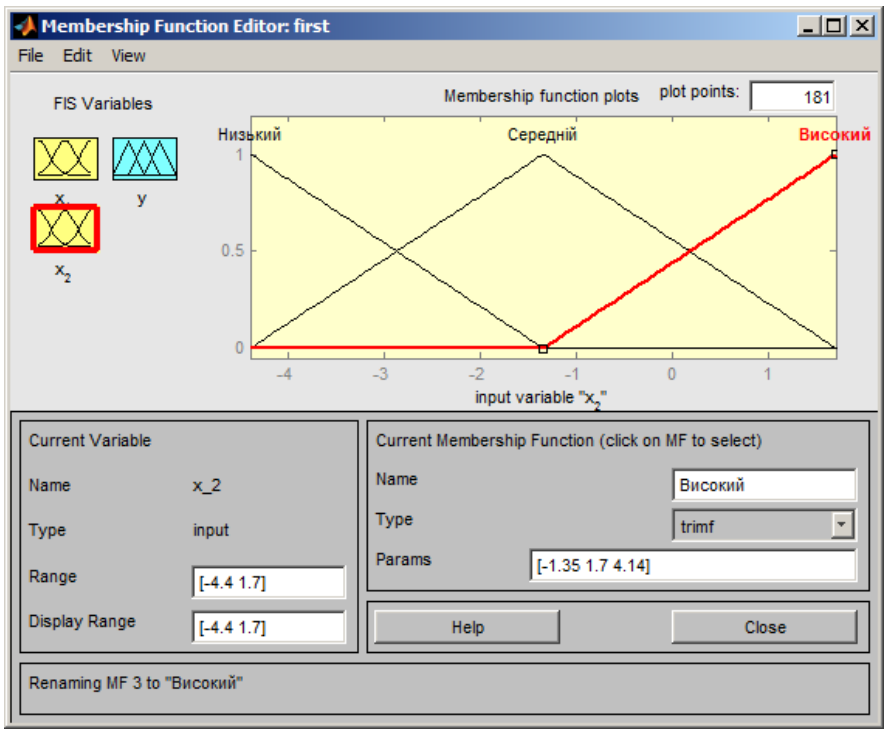


Рисунок 16 – Функції належності змінної x_2

Крок 13. Задамо функції належності змінної y. Для лінгвістичної оцінки цієї змінної використаємо 5 термів з гаусовими функціями належності. Для цього активуємо вихідну змінну натиснувши ліву кнопку миші на блоці y. Задамо діапазон зміни змінної y. Для цього надрукуємо `50 50` в полі Range і натиснемо <Enter>. В меню Edit оберемо команду Add MFs.... В новому діалоговому вікні оберемо в полі MF type тип функції належності gaussmf, а в полі Number of MFs – 5 термів. Після чого натискаємо <Enter>.

Крок 14. За аналогією з кроком 10 задамо наступні найменування термів змінної y: “Низький”, “Нижче середнього”, “Середній”, “Вище середнього” та “Високий”. В результаті отримуємо графічне вікно з рис. 17.

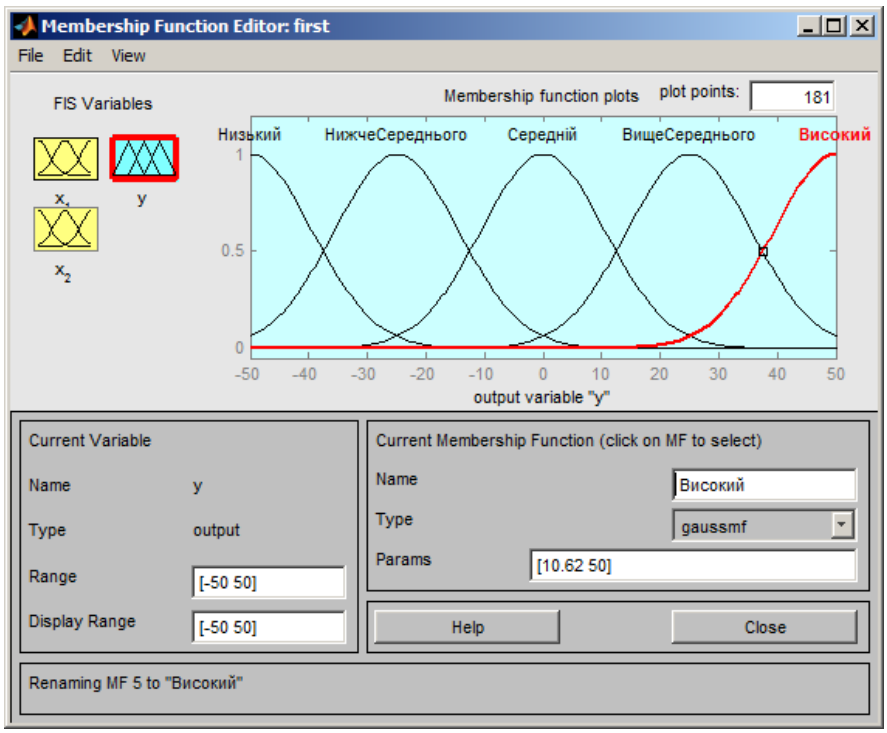


Рисунок 17 – Функції належності змінної y

Крок 15. Перейдемо в редактор бази знань – в RuleEditor. Для цього оберемо в меню Edit команду Rules.

Крок 16. На основі візуального спостереження за графіком з рис. 13, сформуємо наступні 7 нечітких правил:

якщо x_1=Низький та x_2=Низький, тоді y=Високий;

якщо x_1=Низький та x_2=Середній, тоді y=Низький;

якщо x_1=Низький та x_2=Високий, тоді y=Високий;

якщо x_1=Середній, тоді y=Середній;

якщо x_1=Високий і x_2=Низький, тоді y=Вище середнього;

якщо x_1=Високий та x_2=Середній, тоді y=Нижче середнього;

якщо x_1=Високий та x_2=Високий, тоді y=Вище середнього.

Для введення правила оберемо в меню відповідну комбінацію термів і натиснемо кнопку Add rule. На рис. 18 зображено вікно редактора бази знань після введення усіх 7 правил. В кінці кожного правила в дужках наведено ваговий коефіцієнт. Ваговий коефіцієнт можна змінити в полі Weight.

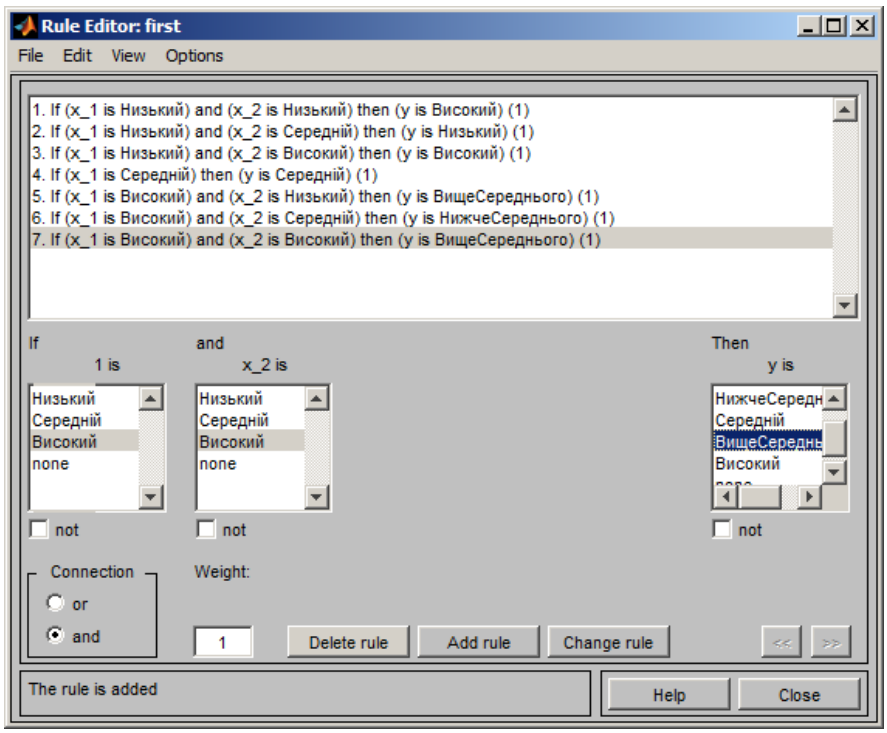


Рисунок 18 – Вікно нечітких правил

Крок 17. Збережемо створену нечітку систему. Для цього в меню File в підменю Export оберемо команду To File.... На диску нечітка система зберігається у формі текстового файлу нескладної структури. Зміст цього файлу наведено нижче.

```
[System]
Name='first'
Type='mamdani'
Version=2.0
NumInputs=2
NumOutputs=1
NumRules=7
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='x_1'
Range=[-7 3]
NumMFs=3
MF1='Низький':'trimf',[-11 -7 -3]
```

```
MF2='Середній':'trimf',[-6 -2 2]
MF3='Високий':'trimf',[-1 3 7]

[Input2]
Name='x_2'
Range=[-4.4 1.7]
NumMFs=3
MF1='Низький': 'trimf',[-6.84 -4.4 -1.35]
MF2='Середній':'trimf',[-4.4 -1.35 1.7]
MF3='Високий':'trimf',[-1.35 1.7 4.14]

[Output1]
Name='y'
Range=[-50 50]
NumMFs=5
MF1='Низький':'gaussmf',[10.62 -50]
MF2='НижчеСерднього':'gaussmf',[10.62 -25]
MF3='Середній':'gaussmf',[10.62 -2.22e-16]
MF4='ВищеСереднього':'gaussmf',[10.62 25]
MF5='Високий':'gaussmf',[10.62 50]

[Rules]
1 1, 5 (1) : 1
1 2, 1 (1) : 1
1 3, 5 (1) : 1
2 0, 3 (1) : 1
3 1, 4 (1) : 1
3 2, 2 (1) : 1
3 3, 4 (1) : 1
```

На рис. 19 наведено вікно візуалізації нечіткого виведення для поточних значень $x_1 = -4$ та $x_2 = 1$. Воно активується командою Rules меню View. В полі Input вказані значення вхідних змінних, для яких виконується логічне виведення. Під час виведення чітке вхідне значення $x_1 = -4$ перетворюється в бінечітку множину $\tilde{\tilde{x}}_1 = \left(\frac{0.25}{Низький}, \frac{0.5}{Середній}, \frac{0}{Високий} \right)$, а вхідне значення $x_2 = 1$ перетворюється в бінечітку множину $\tilde{\tilde{x}}_2 = \left(\frac{0}{Низький}, \frac{0.23}{Середній}, \frac{0.77}{Високий} \right)$. На рис. 19 ступені належності термам вхідних змінних показано червоними числами на осі ординат та

жовтою заливкою на графіках. За цих бінечітких вхідних значень ступені виконання правил становлять:

$$\mu_1 = \min(0.25, 0) = 0;$$

$$\mu_2 = \min(0.25, 0.23) = 0.23;$$

$$\mu_3 = \min(0.25, 0.77) = 0.25;$$

$$\mu_4 = \min(0.5, 1) = 0.5;$$

$$\mu_5 = \min(0, 0) = 0;$$

$$\mu_6 = \min(0, 0.23) = 0;$$

$$\mu_7 = \min(0, 0.77) = 0.$$

Нечіткий висновок за кожним правилом зображено синьою заливкою графіків функцій належності термів змінної *y*. Результуюче нечітке значення знайдено об'єднанням висновків за другим, третім та четвертими правилами. Воно зображено в нижньому правому куті рис. 19. Там же широкою червоною лінією зображено результат дефаззифікації – $y = 0.534$.

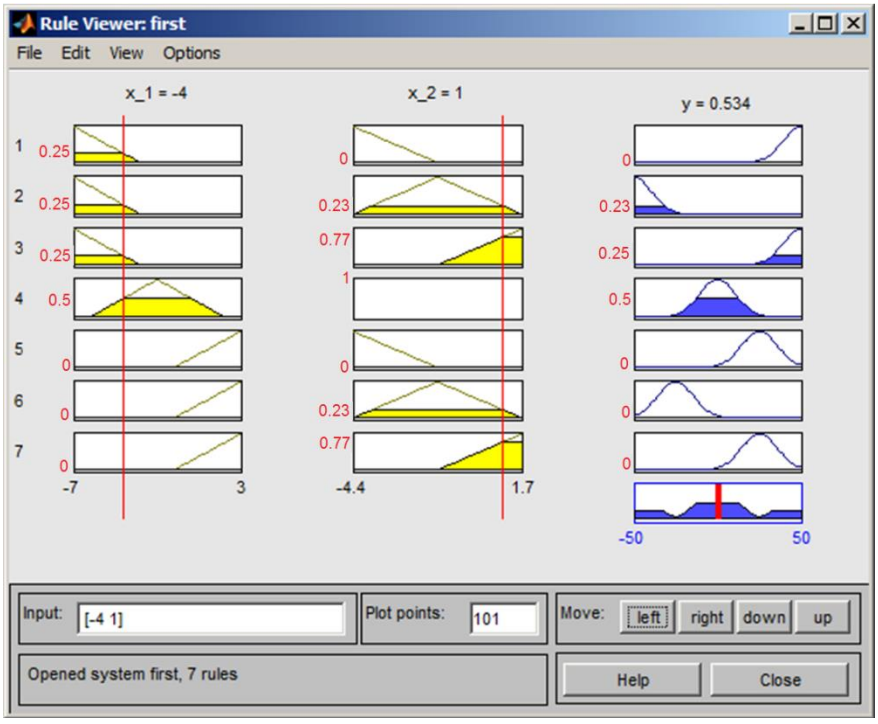


Рисунок 19 – Візуалізація нечіткого виведення

На рис. 20 зображена поверхня "входи – вихід", яка відповідає синтезованій нечіткій системі. Для виводу цього вікна необхідно в меню View обрати команду Surface. Порівнюючи поверхні на рис. 13 і рис. 20,

робимо висновок, що 7 нечітких правил досить добре описують складну нелінійну залежність.

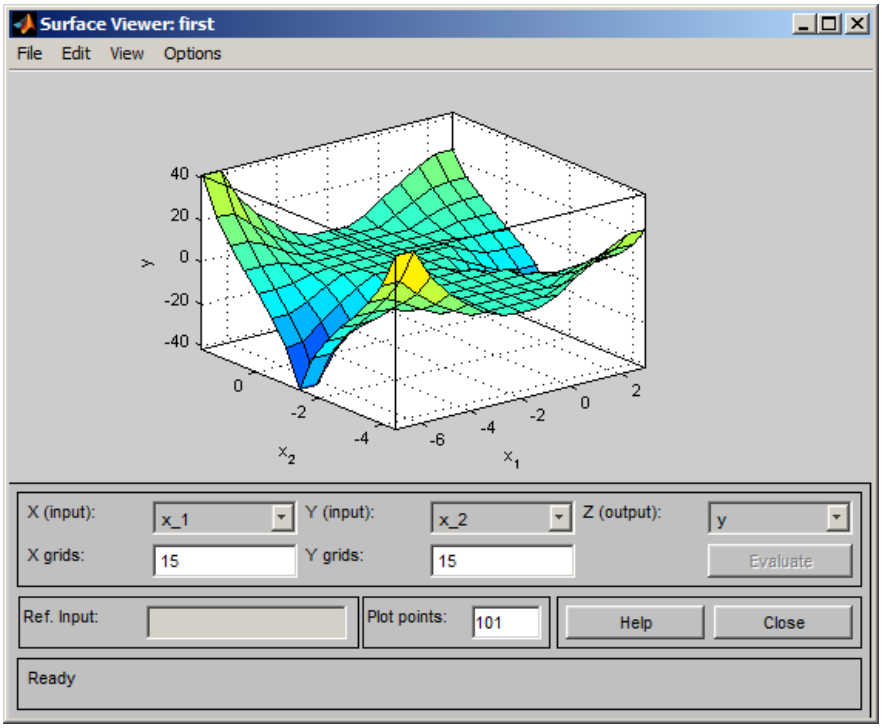


Рисунок 20 – Поверхня “входи – вихід” нечіткої системи

Третє завдання можна виконати рівномірно розподіливши тестові точки по факторному простору.

Для виконання *четвертого*, *п’ятого* чи *шостого завдань* недоцільно застосовувати громіздкі конструкції з типовою функцією додавання правила `addrule`. Краще записати до нечіткої бази знань усі можливі правила та активувати їх за допомогою вагових коефіцієнтів. Якщо занулити ваговий коефіцієнт, тоді відповідне правило не впливатиме на логічне виведення. Це еквівалентно вилученню правила з бази знань. Для активації правила присвоїмо ваговому коефіцієнту одиничне значення. Поле вагового коефіцієнта *i*-го правила нечіткої бази знань, яка представлена структурою `fis`, вказується так: `fis.rule(i).weight`. Розгорнута схема fis-структури наведена на рис. 21.

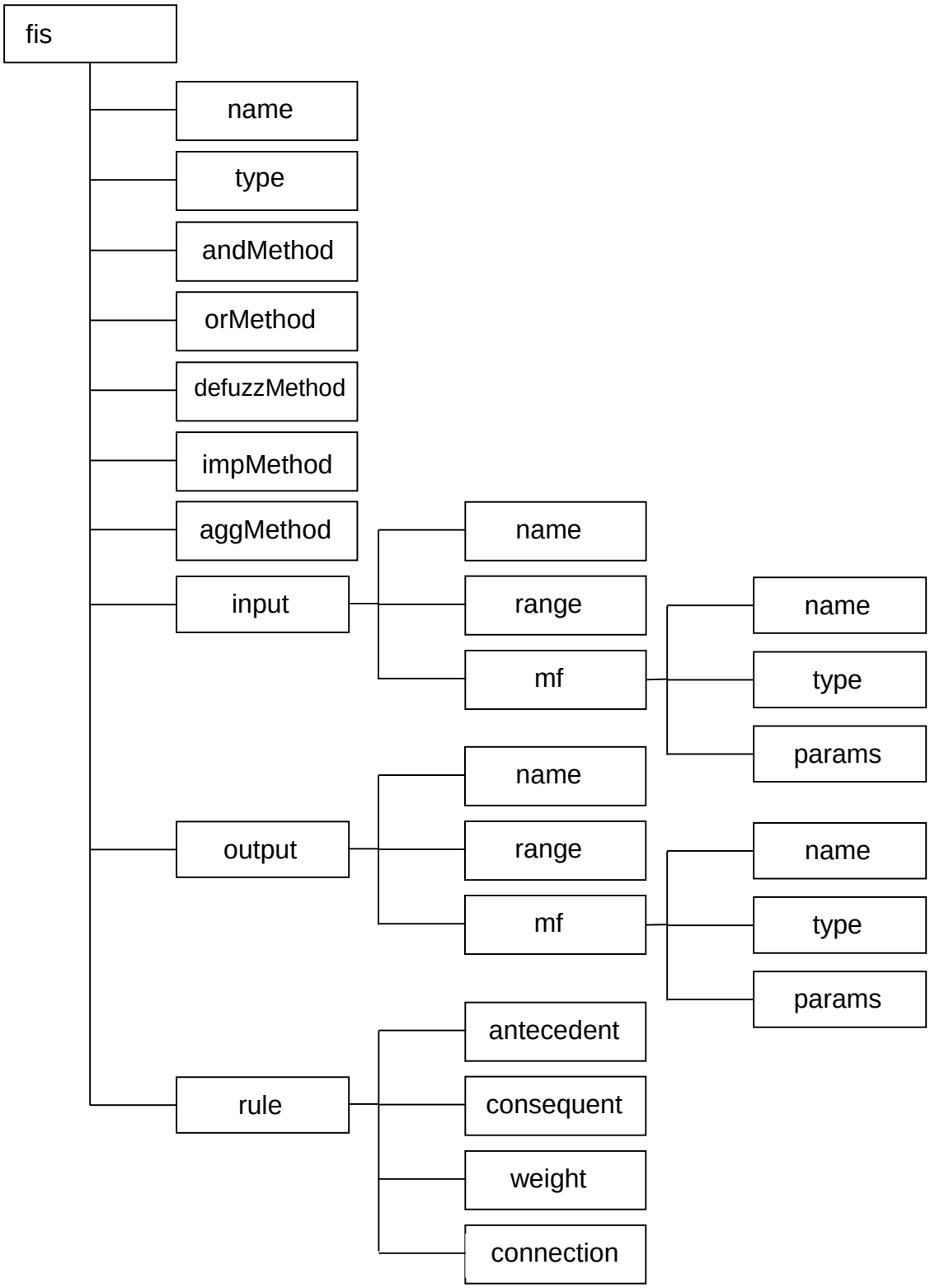


Рисунок 21– Fis-структура

Поля fis-структури призначені для збереження такої інформації:

`name` – найменування системи нечіткого логічного виведення;

type – тип системи (запрограмовані реалізації Mamdani та Sugeno);

andMethod – реалізація логічної операції ТА (запрограмовані реалізації: min – мінімум та prod – добуток);

orMethod – реалізація логічної операції АБО (запрограмовані реалізації: max – максимум та probor – ймовірнісне АБО);

defuzzMethod – метод дефаззифікації (запрограмовані методи для систем типу Мамдані: centroid – центр тяжіння; bisector – медіана; lom – найбільше з максимумів; som – найменше з максимумів; mom – середнє з максимумів);

impMethod – реалізація імплікації (запрограмовані реалізації: min – мінімум та prod – добуток);

aggMethod – реалізація агрегування (запрограмовані реалізації: max – максимум; sum – сума та probor – ймовірнісне АБО);

input – масив вхідних змінних;

input.name – найменування вхідної змінної;

input.range – діапазон зміни вхідної змінної;

input.mf – масив функцій належності вхідної змінної;

input.mf.name – терм вхідної змінної;

input.mf.type – тип функції належності вхідної змінної (запрограмовані функції належності: dsigmf – різниця двох сигмоїдних функцій; gauss2mf – двобічна гаусова; gaussmf – гаусова; gbellmf – узагальнена дзвонова; pimf – пі-подібна; psigmf – добуток двох сигмоїдних функцій; sigmf – сигмоїдна; smf – s-подібна; trapmf – трапецієва; trimf – трикутна; zmf – z-подібна);

input.mf.params – масив параметрів функцій належності вхідної змінної;

output – масив вихідних змінних;

output.name – найменування вихідної змінної;

output.range – діапазон зміни вихідної змінної;

output.mf – масив функцій належності вихідної змінної;

output.mf.name – терм вихідної змінної;

output.mf.type – тип функції належності вихідної змінної (запрограмовані функції належності для бази знань Мамдані: dsigmf – різниця двох сигмоїдних функцій; gauss2mf – двобічна гаусова;

`gaussmf` – гаусова; `gbellmf` – узагальнена дзвонова; `pimf` – пі-подібна; `psigmf` – добуток двох сигмоїдних функцій; `sigmf` – сигмоїдна; `smf` – s-подібна; `trapmf` – трапецієва; `trimf` – трикутна; `zmf` – z-подібна);

`output.mf.params` – масив параметрів функції належності вихідної змінної;

`rule` – масив правил нечіткої бази знань;

`rule.antecedent` – антецедент правила у формі переліку номерів термів у порядку запису вхідних змінних. Число 0 вказує терм “Don’t care”. Від'ємне значення вказує на використання замість відповідної нечіткої множини її доповнення. Наприклад, якщо першим термом є “Низький”, тоді число “–1” означає використання терму “Не низький”.

`rule.consequent` – консеквент правила у формі переліку номерів термів вихідних змінних. Число 0 вказує, що правило не поширюється на відповідну вихідну змінну. Від'ємне значення вказує на доповнення;

`rule.weight` – ваговий коефіцієнт правила. Задається числом з діапазону [0, 1];

`rule.connection` – логічне зв’язка фрагментів антецедента правила: 1 – логічне ТА; 2 – логічне АБО.

Виконання *шостого* завдання для великих баз знань потребує значних обчислювальних ресурсів. Для бази знань з 16-ти правил перебір усіх варіантів за тестової вибірки з 100 точок займе біля 10 хвилин. Якщо додати ще 3 правила, тоді тривалість експерименту може зрости до кількох годин. Якщо додати ще 5 правил експеримент триватиме кілька днів.

Для повного перебору правил нечіткої бази знань можна скористатись функцією комбінаторики `nchoosek` наступним чином:

```
%RMSEfis - функція розрахунку RMSE, яку слід створити
%власноруч.
%INP - вхідні значення тестової вибірки.
%OUT - вихідні значення тестової вибірки.
fis = readfis('first');    % зчитуємо базу знань first.fis.
Nrules = length(fis.rule);
for k=1:Nrules
    % Кількість комбінацій:
    N=nchoosek(Nrules,k);     .
    % Матриця комбінацій із k правил:
    M=nchoosek(1:Nrules,k);
    Error{k} = zeros(N,1);
```

```
    for i=1:N
        fis.rule(:).weight=0;
        fis.rule( M(i,:) ).weight=1;
        Error{k}(i)=RMSEfis(fis, INP, OUT);
    end
end
```

Якщо кількість правил-кандидатів дорівнює N, тоді можна згенерувати $(2^N - 1)$ унікальних баз знань. Усі ці комбінації правил можна отримати за допомогою швидких операцій побітового зсуву `bitshift` та перевірки по модулю 2 `mod`.

Криву навчання можна побудувати і за жадібним алгоритмом, який полягає у почерговому додаванні найкращих правил. Алгоритм стартує з порожньої бази знань і на першій ітерації додає одне правило з мінімальною нев'язкою. На кожній наступній ітерації серед правил, що залишились, обирається те, додавання якого забезпечить найменшу нев'язку нечіткої бази знань. Приклад кривих навчання для бази знань наведено на рис. 22. Найкращою виявилась база знань з трьох нечітких правил з номерами 1, 2 та 4. В звіті на кривих навчання слід виділити найточнішу базу знань та вказати, з яких правил вона складається.

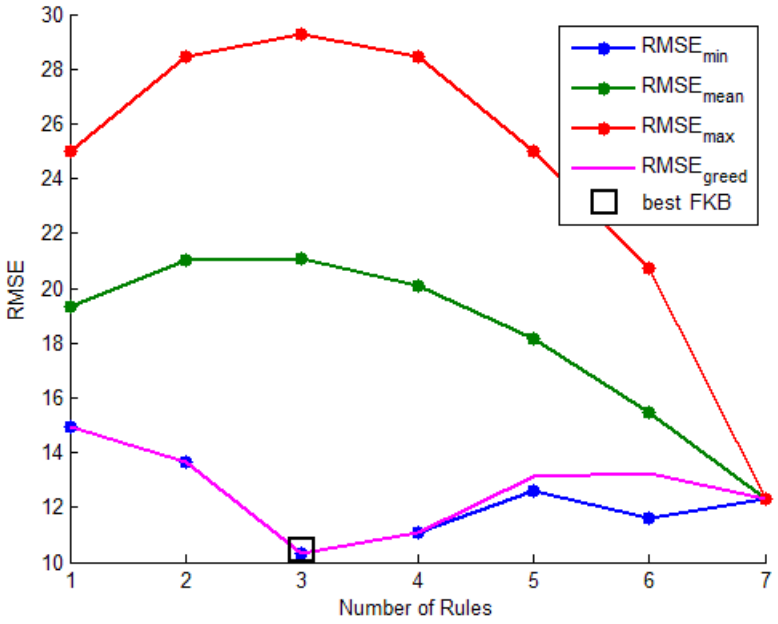


Рисунок 22 – Криві навчання бази знань Мамдані з рис. 18

Для відображення в звіті функцій належності входів та виходу (рис. 23) можна викликати функцію `plotmf`.

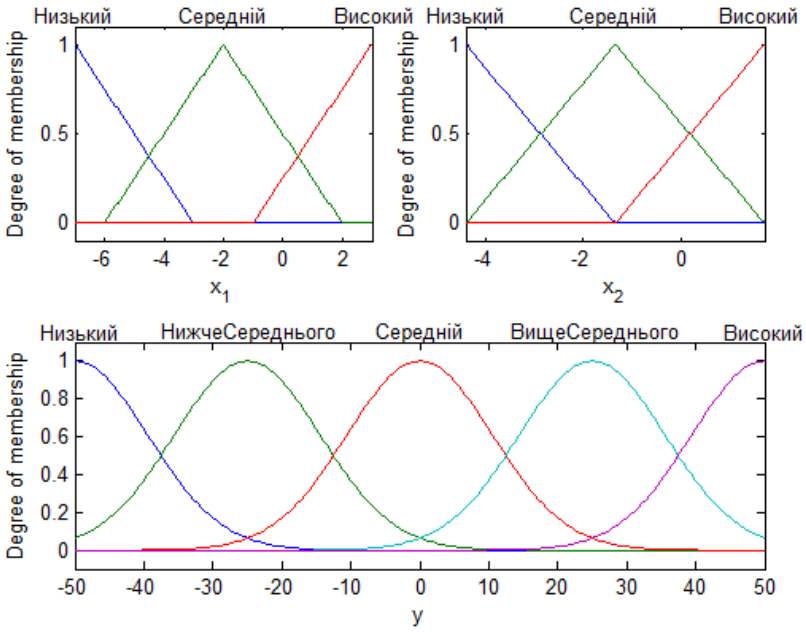


Рисунок 23 – Графіки функцій належності входів бази знань Мамдані

Під час виконання лабораторної роботи доцільно використовувати такі функції:

`fuzzy`	–	виклик fis-редактору;
`readfis`	–	зчитування збереженої нечіткої бази знань;
`setfis`	–	зміна окремих параметрів бази знань;
`evalfis`	–	нечітке логічне виведення за базою знань;
`plotmf`	–	побудова графіків функцій належності термів певної змінної;
`plotfis`	–	графічне представлення структури та основних параметрів системи нечіткого виведення;
`showrule`	–	вивід списку правил бази знань;
`surf`	–	побудова поверхонь;
`unique`	–	вилучення з масиву дублюючих елементів;
`min`	–	значення та порядковий номер мінімального елементу масиву;
`max`	–	значення та порядковий номер максимального елементу масиву;

`find` – знаходження номерів елементів масива, які задовольняють деяку умову;

`reshape` – зміна розміру матриці;

`setdiff` – різниця множин;

`nchoosek` – генерування комбінаторних комбінацій елементів.

Детальний опис функцій `evalfis`, `readfis`, `showrule`, `plotmf` та `plotfis` наведено в Довіднику ключових функцій.

Питання для самоконтролю

1. Наведіть приклади нечітких множин.
2. Представте нечіткими множинами такі висловлювання:
   - висока швидкість потяга;
   - дорога сукня;
   - висока тактова частота;
   - вимогливий викладач;
   - низька похибка вимірювання.
3. В чому особливості нечіткої бази знань?
4. Наведіть приклади лінгвістичних змінних.
5. Чим відрізняється логічне виведення від нечіткого логічного виведення?
6. Наведіть приклад правила з нечіткої бази знань.
7. Наведіть змістовну та формалізовану постановки задачі про рюкзак.
8. Що таке “плато насичення” кривої навчання?
9. Чому навіть за максимальної кількості правил в базі знань нев’язка не дорівнює нулю?
10. За яких умов задачу відбору правил до нечіткої бази знань доцільно вирішувати методом повного перебору?
11. Яка залежність тривалості лабораторного дослідження від кількості правил?
12. На скільки зросте тривалість лабораторного дослідження якщо збільшити тестову вибірку з 100 до 1000 точок?

Лабораторна робота №3

Ідентифікація залежностей за допомогою нечіткого класифікатора

Мета – на основі експериментальних даних побудувати нечітку базу, за якою здійснюється достовірна класифікація.

Теоретичні відомості

Нечіткий класифікатор являє собою відображення $X=(x_1,x_2,\ldots,x_n) \rightarrow y \in \{l_1, l_2, \ldots, l_m\}$ на основі бази нечітких правил. Базу правил нечіткого класифікатора запишемо так:

$$\text{Якщо } (x_1=\tilde{a}_{1j} \text{ та } x_2=\tilde{a}_{2j} \text{ та } \ldots \text{ та } x_n=\tilde{a}_{nj} \text{ з вагою } w_j), \text{ тоді } y=d_j, \quad j=\overline{1,N}, \tag{11}$$

де N – кількість правил;

$d_j \in \{l_1, l_2, \ldots, l_m\}$ – категоріальне значення консеквента j-го правила;

$w_j \in [0,1]$ – ваговий коефіцієнт, який задає достовірність j-го правила, $j=\overline{1,N}$;

$\tilde{a}_{ij}$ – нечіткий терм, яким оцінюється атрибут x_i в j-му правилі, $i=\overline{1,n}$, $j=\overline{1,N}$.

Класифікація поточного об’єкта з атрибутами $X^*=\left(x_1^*, x_2^*, \ldots, x_n^*\right)$ здійснюється таким чином. Спочатку розраховується ступінь виконання j-го правила з (11):

$$\mu_j(X^*) = w_j \cdot \left(\mu_j(x_1^*) \wedge \mu_j(x_2^*) \wedge ... \wedge \ \mu_j(x_n^*) \right), \quad j = \overline{1, N}, \qquad (12)$$

де $\mu_j(x_i^*)$ – ступінь належності значення x_i^* нечіткому терму $\tilde{a}_{ij}$;

$\wedge$ – t-норма, яку зазвичай реалізують операцією мінімуму або добутком.

Ступінь належності вхідного вектору X^* до класів $l_1, l_2, ..., l_m$ розраховується так:

$$\mu_{l_s}(y^*) = \underset{\forall j: d_j = l_s}{agg} \left(\mu_j(X^*) \right), \quad s = \overline{1, m}, \qquad (13)$$

де *agg* – агрегування нечітких висновків за окремими правилами.

В нечітких класифікаторів агрегування логічних висновків за усіма правилами бази знань здійснюють за двома схемами. За першою схемою з єдиним правилом-переможцем (single winner rule) результатом виведення обирається консеквент правила з максимальним ступенем виконання. За другою схемою з голосуючими правилами (voting rules) результатом виведення обирається клас з максимальною сумарною належністю за усіма правилами. Перевагою схеми з єдиним правилом-переможцем є більш інтерпретабельний алгоритм виведення, а схеми з голосуючими правилами – більш гладкі границі розділу між класами в просторі атрибутів.

За агрегування з схемою з єдиним правилом переможцем формула (13) перетвориться на таку:

$$\mu_{l_s}(y^*) = \max_{\forall j: d_j = l_s} \left(\mu_j(X^*) \right), \quad s = \overline{1, m}. \qquad (14)$$

За агрегування з схемою голосування формула (13) перетвориться на таку:

$$\mu_{l_s}(y^*) = \frac{\sum\limits_{\forall j: d_j = l_s, \ j = \overline{1, N}} \mu_j(X^*)}{\max\limits_{s = \overline{1, m}} \left(\sum\limits_{\forall j: d_j = l_s, \ j = \overline{1, N}} \mu_j(X^*) \right)}, \quad s = \overline{1, m}. \qquad (15)$$

Нечітким рішенням задачі класифікації буде нечітка множина

$$\tilde{y}^* = \left(\frac{\mu_{l_1}(y^*)}{l_1}, \frac{\mu_{l_2}(y^*)}{l_2}, ..., \frac{\mu_{l_m}(y^*)}{l_m} \right). \qquad (16)$$

Результатом виведення оберемо ядро нечіткої множини (13), тобто клас з максимальним ступенем належності:

$$y^* = \underset{\{l_1, l_2, ..., l_m\}}{\arg} \quad \underset{s=1, m}{\max} \left(\mu_{l_s}(y^*) \right).$$

Для прикладу розглянемо залежність безпомилковості діяльності людини-оператора від завантаженості та тривалості роботи. З ергономіки відомо, що як замале, так і завелике завантаження зменшують надійність. Під час звичайної діяльності функціональний стан оператора проходить 5 фаз: 1) початкова реакція; 2) гіперкомпенсація; 3) компенсація; 4) субкомпенсація; 5) декомпенсація. Перша та друга фази відповідають приробці, третя – ефективній роботі, четверта і п'ята – втомі. В табл. 8 наводиться нечітка база знань, яка формалізує наведені факти. На рис. 24 наведено результати нечіткого моделювання надійності оператора для 1000 різних комбінацій значень факторів впливу. Агрегування реалізовано операцією максимума, а t-норма – мінімумом. Класифікація відбувається за схемою з єдиним правилом-переможцем.

Таблиця 8 – Нечітка класифікаційна база знань

Завантаження оператора	Тривалість роботи	Ймовірність безпомилкової роботи
Середнє	Компенсація	Висока
Дуже велике	Декомпенсація	Низька
Велике	Початок	Низька
Дуже велике	Субомпенсація	Низька
Дуже велике	Компенсація	Середня
Велике	Субомпенсація	Середня
Мале	Декомпенсація	Середня
Дуже мале	Компенсація	Середня
Середнє	Початок	Середня
Велике	Гіперкомпенсація	Середня

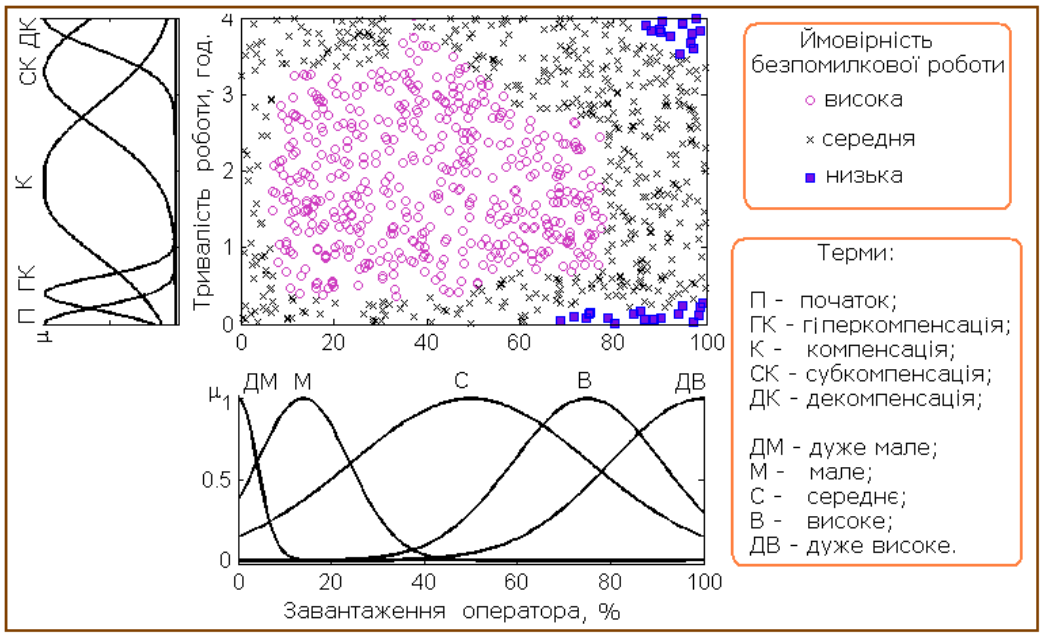


Рисунок 24 – Класифікація за нечіткою базою знань з табл. 8

Для підвищення безпомилковості нечіткий класифікатор навчають по експериментальним даним. Для цього змінюють його параметри, щоб мінімізувати відстань між експериментальними даними та результатами нечіткого виведення. Запишемо навчальну вибірку з M пар "входи – вихід" таким чином:

$$(X_r, y_r), \quad r = \overline{1, M}, \tag{17}$$

де $X_r = (x_{r1}, x_{r2}, \ldots, x_{rn})$ - вхідні атрибути r-го об'єкту;

$y_r \in \{l_1, l_2, \ldots, l_m\}$ - клас r-го об'єкту.

Параметри функцій належності термів з бази знань (11) зведемо у вектор P, а вагові коефіцієнти у вектор W. Позначимо через $F(K, X_r) \in \{l_1, l_2, \ldots, l_m\}$ – результат класифікації за базою знань з параметрами $K = (P, W)$ для вхідного вектору X_r з r-го рядка вибірки (17).

Навчання нечіткого класифікатора полягає в знаходженні такого вектору K, який мінімізує частоту помилок класифікації на тестовій вибірці. При цьому для налаштування параметрів K використовується лише навчальна вибірка (17). Навчання розглядається як задача оптимізації із пошуку таких керованих змінних K, які мінімізують відстань між результатами логічного виведення та експериментальними даними з

вибірки (17). Цю відстань, яку назвемо критерієм навчання, можна визначити у різний спосіб.

Критерій 1 – частота помилок класифікації:

$$Crit_1 = \frac{1}{M} \sum_{r=\overline{1,M}} \Delta_r(K), \tag{18}$$

де $\Delta_r(K) = \begin{cases} 1, & якщо \ \ y_r \neq F(K, X_r) \\ 0, & якщо \ \ y_r = F(K, X_r) \end{cases}$.

Переваги критерію полягають в його простоті та ясній змістовній інтерпретації. Але цільова функція в задачі оптимізації за цим критерієм приймає дискретні значення, що ускладнює застосування швидких градієнтних методів оптимізації, особливо за малих вибірок даних.

Критерій 2 – квадратична нев'язка між двома нечіткими множинами – бажаними та реальними результатами класифікації. Для її розрахунку значення вихідної змінної y в навчальній вибірці фаззифікують таким чином:

$$\left.\begin{array}{ll} \tilde{y} = \left(\frac{1}{l_1}, \frac{0}{l_2}, ..., \frac{0}{l_m}\right), & якщо \ \ y = l_1 \\ \tilde{y} = \left(\frac{0}{l_1}, \frac{1}{l_2}, ..., \frac{0}{l_m}\right), & якщо \ \ y = l_2 \\ \vdots & \\ \tilde{y} = \left(\frac{0}{l_1}, \frac{0}{l_2}, ..., \frac{1}{l_m}\right), & якщо \ \ y = l_m \end{array}\right\}. \tag{19}$$

Критерій навчання враховує відстань між висновком у формі нечіткої множини (16) та бажаним нечітким значенням вихідної змінної (19):

$$Crit_2 = \sum_{r=\overline{1,M}} D_r(K), \tag{20}$$

де $D_r(K)$ – відстань між бажаною та дійсною вихідними нечіткими множинами при класифікації r-го об'єкту з навчальної вибірки (17).

Для розрахунку $D_r(K)$ використовується евклідова метрика:

$$D_r(K) = \sum_{s=1,\,m} \left(\mu_{l_s}(y_r) - \mu_{l_s}(K, X_r)\right)^2, \qquad (21)$$

де $\mu_{l_s}(y_r)$ – ступінь належності r-го об'єкту навчальної вибірки до класу l_s згідно до (19);

$\mu_{l_s}(K, X_r)$ – розрахований за (15) ступінь належності висновку за нечіткою моделлю з параметрами K до класу l_s для вхідного вектору X_r.

Для прикладу розрахуємо відстань (21) за результатами логічного виведення з рис. 25. У випадку правильної класифікації (рис. 25а) відстань дорівнює:

$$D_a = (0-0.1)^2 + (0-0.2)^2 + (1-0.8)^2 + (0-0.5)^2 = 0.34.$$

У випадку помилкової класифікації (рис. 25б) відстань дорівнює:

$$D_б = (0-0.1)^2 + (1-0.2)^2 + (0-0.8)^2 + (0-0.5)^2 = 1.54.$$

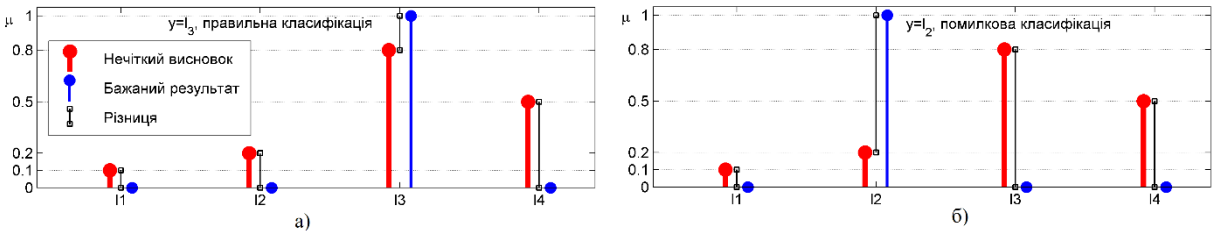


Рисунок 25 – До формули (21)

а) правильна класифікація; б) помилкова класифікація

Критерій 3 – квадратична нев'язка між нечіткими бажаними та реальними результатами класифікації з додатковим штрафом за помилкове рішення. Цей критерій успадковує переваги двох попередніх. Ідея полягає в збільшенні відстані D для помилково класифікованих об'єктів:

- $$Crit_3 = \sum_{r=1,\,M} (\Delta_r(K) \cdot p + 1) \cdot D_r(K), \qquad (22)$$

де $p>0$– штрафний коефіцієнт.

Під час навчання за критерієм (22) вибір напрямку крокування до оптимуму найбільшою мірою залежить від помилково класифікованих

об’єктів. Така поведінка схожа на адаптивний метод оптимізації Л. Растригіна, коли для повторного навчання частіше пред’являють помилково розпізнані об’єкти.

Критерій 4 – відстань між головними конкурентами з штрафом за помилкове рішення. Ідея цього критерію полягає у врахуванні різниці належностей нечіткого висновку лише до головних конкурентів. За алгоритмом логічного виведення рішенням обирається клас з максимальним ступенем належності. Позначимо цей клас-переможець через *win* та присвоїмо йому перший ранг. У випадку правильної класифікації головним конкурентом прийнятого рішення є *vicewin* – клас з другим рангом, тобто клас з другим за величиною ступенем належності (рис. 26а). Чим більша різниця між ступенями належності до класів *win* та *vicewin*, тим більша впевненість у логічному висновку, і тим далі об’єкт знаходиться від границі розділу класів. Позначимо через smax - операцію знаходження другого за величною елемента множини. Тоді, для *r*-го об’єкту з вибірки (17):

$$\mu_{win}(X_r) = \max\limits_{s=1,m}(\mu_{l_s}(X_r));$$

$$\mu_{vicewin}(X_r) = \operatorname{smax}\limits_{s=1,m}(\mu_{l_s}(X_r)).$$

Відповідно, різниця між головними конкурентами дорівнює $\mu_{win}(X_r) - \mu_{vicewin}(X_r)$.

За неправильної класифікації помилково прийняте рішення буде головним конкурентом правильного класу (рис. 26б). Відповідно, бажано зменшити різницю між ступенями належності до помилкового рішення та до правильного класу. Різницю між головними конкурентами в цьому випадку запишемо так: $\mu_{win}(X_r) - \mu_{y_r}(X_r)$.

В критерії навчання враховуємо відносні показники, розділивши різницю на ступінь належності класу-переможцю. За правильної класифікації відносна різниця дорівнює $D_r^1 = \dfrac{\mu_{win}(X_r) - \mu_{vicewin}(X_r)}{\mu_{win}(X_r)}$, а за неправильної – $D_r^0 = \dfrac{\mu_{win}(X_r) - \mu_{y_r}(X_r)}{\mu_{win}(X_r)}$. Крім того, аналогічно до

критерія 3, за помилкової класифікації зважимо різницю штрафним коефіцієнтом. Математично критерій навчання запишемо таким чином:

$$Crit_4 = p \cdot \sum_{\substack{y_r \neq F(K, X_r) \\ r=\overline{1, M}}} D_r^0(K) - \sum_{\substack{y_r = F(K, X_r) \\ r=\overline{1, M}}} D_r^1(K), \tag{23}$$

де $p \geq 1$ – штрафний коефіцієнт.

Для прикладу розрахуємо відстань (23) за результатами нечіткого виведення з рис. 26. За правильної класифікації (див. рис. 26а) відстань дорівнює:

$$D_a^1 = \frac{0.8 - 0.5}{0.8} = 0.375 .$$

У випадку помилкової класифікації (рис. 26б) за штрафного коефіцієнту p=3 відстань дорівнює:

$$D_б^0 = 3 \cdot \frac{0.8 - 0.2}{0.8} = 2.25 .$$

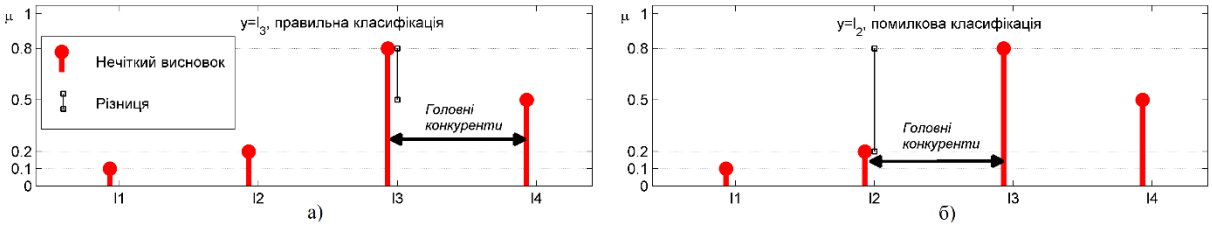


Рисунок 26 – Головні конкуренти

Критерій 5 – квадратична відстань між головними конкурентами з штрафом за помилкове рішення. Цей критерій є модифікацією попереднього. Відмінність полягає у використанні не абсолютних відстаней, а їх квадратів:

$$Crit_5 = p \cdot \sum_{\substack{y_r \neq F(K, X_r) \\ r=\overline{1, M}}} D_r^0(K)^2 - \sum_{\substack{y_r = F(K, X_r) \\ r=\overline{1, M}}} D_r^1(K)^2 .$$

Піднесення до квадрату в $Crit_5$ дозволяє, так само, як в методі найменших квадратів, збільшити в критерії навчання вплив великих різниць та зневілювати вплив малих різниць.

Завдання на лабораторну роботу

В лабораторній роботі проектується нечіткий класифікатор, який моделює залежність для однієї із задач репозитарію автоматичного навчання Каліфорнійського університету в Ірвіні [22]. Задача обирається згідно варіанту з табл. 9. Лабораторна робота полягає у виконанні таких завдань.

1. Сформулювати змістовну постановку задачі згідно варіанту. Дослідження проводити лише для кількісних вхідних атрибутів.

2. Розбити дані на навчальну та тестову вибірки та перевірити їх репрезентативність.

3. Побудувати невелике дерево рішень та на його основі сформулювати правила нечіткого класифікатора з 2–3 вхідними змінними.

4. Навчити нечіткий класифікатор шляхом налаштування вагових коефіцієнтів правил та параметрів функцій належності нечітких термів.

5. Порівняти створений класифікатор з результатами вирішення цієї задачі іншими дослідниками.

Таблиця 9 – Варіанти завдання

Варіант	Задача
1	Activity Recognition from Single Chest-Mounted Accelerometer
2	Bank Marketing
3	Banknote authentication
4	Car Evaluation Data Set
5	Cardiotocography
6	Climate Model Simulation Crashes
7	Credit Approval
8	Cylinder Bands
9	Ecoli
10	Glass Identification
11	Image Segmentation
12	MAGIC Gamma Telescope
13	Nursery
14	Page Blocks Classification
15	Pen-Based Recognition of Handwritten Digits

Варіант	Задача
16	Sensorless Drive Diagnosis
17	Spambase
18	Stalog (Shuttle)
19	Statlog (German Credit Data)
20	Statlog (Vehicle Silhouettes)
21	Steel Plates Faults
22	User Knowledge Modeling Data Set
23	Wilt
24	Yeast

Рекомендації

1. Для поглибленого вивчення матеріалу рекомендуємо літературу [6, 8–11, 13, 18–21, 23, 24].

Під час виконання *першого завдання* слід навести змістовну постановку задачі дослідження. Приклад виконання цього завдання для задачі класифікації вин наведено нижче.

Розглядається тестова задача Wine Dataset з UCI Machine Learning Repository, яка полягає у виявленні сорту винограду (y), з якого виготовлено вино. База даних містить результати лабораторного аналізу за 13-ма показниками 178 зразків італійських вин, виготовлених в одному регіоні. Для кожного зразка вказано 1 із трьох сортів винограду, з якого виготовлено вино. Детальна інформація про задачу знаходиться за посиланням https://archive.ics.uci.edu/ml/datasets/Wine .

При виконанні *другого завдання* в навчальну вибірку можна включити непарні рядки даних, а в тестову – парні. Також бажано, щоб рядки вибірки, що містять крайні значення за кожною ознакою потрапили у навчальну вибірку. За цими правилами навчальна вибірка для задачі розпізнавання вина складається із 100 рядків, а тестова – із 78. Для перевірки репрезентативності вибірок розрахуємо математичне сподівання та дисперсію для кожної вхідної змінної та частоти кожного класу для вихідної змінної. Значення мають бути приблизно однаковими попарно для навчальної та тестової вибірок.

Для виконання *третього завдання* для виявлення інформативних ознак можна побудувати дерево рішень (рис. 27). За цим деревом та за двовимірним розподілом класів з рис. 28 сформуємо 5 правил нечіткого класифікатора (табл. 10).

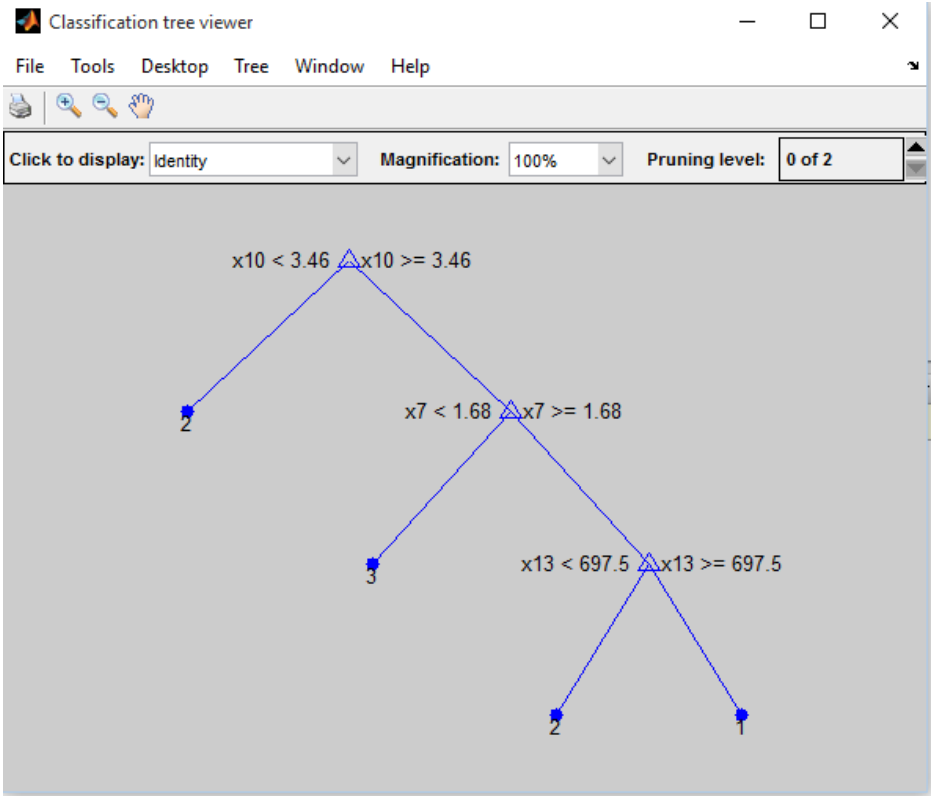


Рисунок 27 – Дерево рішень для задачі розпізнавання вин

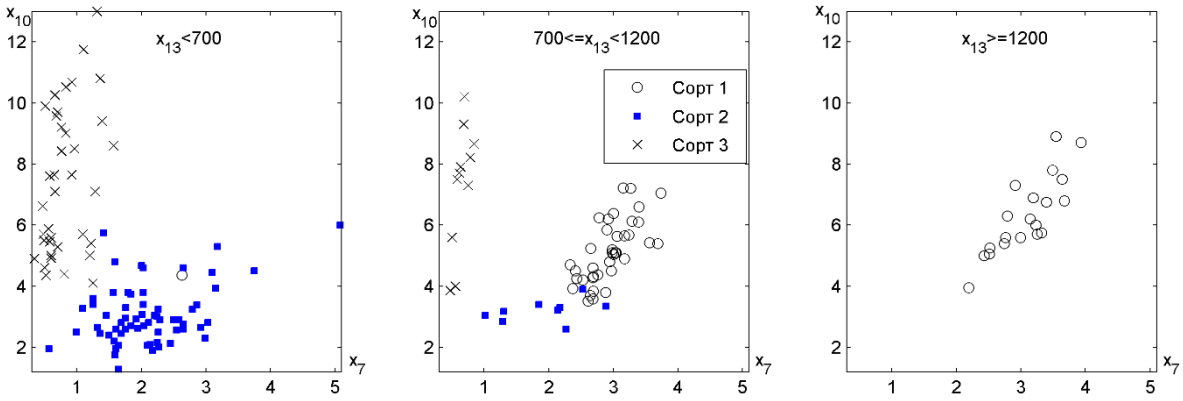


Рисунок 28 – Двовимірні розподіли класів в задачі розпізнавання вин

Таблиця 10 – Нечітка база знань

№	x_7	x_{10}	x_{13}	y	Вага правила	
					до навчання	після навчання
1	–	–	Високий	Сорт 1	1	0.89
2	Високий	Високий	Середній	Сорт 1	1	0.01
3	–	Низький	Низький	Сорт 2	1	0.98
4	Низький	Низький	Середній	Сорт 2	1	0.84
5	Низький	Високий	-	Сорт 3	1	1

Для виконання *четвертого завдання* необхідно уважно прочитати інструкції з підр. 4.3 книги [21]. Справа в тому, що розробники системи MATLAB 7 не запрограмували в пакеті Fuzzy Logic Toolbox логічне виведення для нечіткого класифікатора. Тому, в [21] запропоновано специфічний спосіб програмної реалізації нечіткого класифікатора через базу знань Сугено. Як приклад нижче наведено fis-файл такого нечіткого класифікатора з правилами із табл. 10.

```
[System]
Name='Wine5_7-10-13.fis'
Type='sugeno'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=5
AndMethod='min'
OrMethod='max'
ImpMethod='prod'
AggMethod='sum'
DefuzzMethod='wtaver'

[Input1]
Name='x_7'
Range=[0.34 5.08]
NumMFs=2
MF1='Низький':'gaussmf',[2.33258842015125 0.34]
MF2='Високий':'gaussmf',[2.23376586364764 5.08]

[Input2]
Name='x_1_0'
```

```
Range=[1.28 13]
NumMFs=2
MF1='Низький':'gaussmf',[5.694 1.28]
MF2='Високий':'gaussmf',[5.109 13]

[Input3]
Name='x_1_3'
Range=[278 1680]
NumMFs=3
MF1='Низький':'gaussmf',[244 278]
MF2='Середній':'gaussmf',[342 1071]
MF3='Високий':'gaussmf',[352 1680]

[Output1]
Name='y'
Range=[0 1]
NumMFs=3
MF1='Сорт 1':'constant',[1]
MF2='Сорт 2':'constant',[2]
MF3='Сорт 3':'constant',[3]

[Rules]
0 0 3, 1 (1) : 1
2 2 2, 1 (1) : 1
0 1 1, 2 (1) : 1
1 1 2, 2 (1) : 1
1 2 0, 3 (1) : 1
```

Щоб здійснити класифікацію необхідно застосувати розроблену нами функцію `fuzzy_classifier_v`, лістинг якої наведено нижче.

```
function [decision, mf_grades] =
    fuzzy_classifier_v (x, fis, type, inference)
%FUZZY_CLASSIFIER_V - нечітке виведення для задачі
%класифікації.
%x - вхідний вектор (вхідні атрибути об'єкту класифікації);
%fis - система нечіткого виведення;
%type - формат результату:
%    'number' - порядковий номер класа (default value);
%    'name' - назва класа.
%inference - тип виведення:
%    'single' - single winner rule (default value);
%    'voting' - voting rules
%decision - результат класифікації об'єкта x;
%mf_grades - вектор ступенів належності кожному класу.
```

```
%-------------------------------------------------------------
%Ресурси: Fuzzy Logic Toolbox v.2.X
%Ver. 2.1   11/16/2015
%(c) Serhiy Shtovba and Anastasiia Galushchak,
%    Vinnytsia National Technical University
%    shtovba@ksu.vntu.edu.ua
%    www.shtovba.vinnitsa.com
%=============================================================
%Перевірка вхідних аргументів:
if nargin<2
   error ('Необхідно задати 2, 3 або 4 вхідні аргументи');
end
[tmp1 tmp2]=size(x);
if tmp1~=1
   error ('Класифікація виконується лише для одного об`єкта');
end
if isfis(fis)==0
   error('Другим аргументом має бути нечітка система');
end
if lower(fis.type(1:6))~='sugeno'
   error('Нечітка система має бути типу Сугено');
end
if nargin==2
   type='number';     % < встановлення значення за замовченням
   inference='single';
end
if nargin==3
   inference='single';% < встановлення значення за замовченням
end
if isempty(type)
   type='number';
end
number_of_decisions=length(fis.output(1).mf);
number_of_rules=length(fis.rule);
[a,b,c,d]=evalfis(x,fis);  % < нечітке виведення
maxd=max(d);
if maxd==0
   error('Правила не спрацювали - нульова належність')
end
mf_grades_single(1:number_of_decisions)=0;
mf_grades_voting=mf_grades_single;
for i=1:number_of_rules
    index=fis.rule(i).consequent;%< номер класа в і-му правилі
    %Об'єднуємо операцією максимума чи додаванням ступенів
    %належності до одного і того ж класу за різними правилам:
```

```
    mf_grades_single(index)=max(mf_grades_single(index),d(i));
    mf_grades_voting(index)=mf_grades_voting(index)+d(i);
end
switch inference
    case 'single',
        %Номера правил з максимальним ступенем виконання:
        rule_num_win=
            find(mf_grades_single==max(mf_grades_single));
        if length(rule_num_win)==1
           decision=rule_num_win;
        else
           rule_num_nowin =
               setdiff(1:number_of_decisions, rule_num_win);
           mf_grades_voting(rule_num_nowin)=0;
           [muwin decision]=max(mf_grades_voting);
        end
        mf_grades=mf_grades_single;
    case 'voting',
        %Номера правил з максимальним ступенем виконання:
        rule_num_win=
            find(mf_grades_voting==max(mf_grades_voting));
        if length(rule_num_win)==1
           decision=rule_num_win;
        else
           rule_num_nowin=
               setdiff(1:number_of_decisions, rule_num_win);
           mf_grades_single(rule_num_nowin)=0;
           [muwin decision]=max(mf_grades_single);
        end
        f_grades=mf_grades_voting;
    otherwise,
        error('Некоректий алгоритм нечіткого виведення.
        Допустимі значення 4 аргумента - single та voting')
end
%Повертаємо результат класифікації в необхідному форматі:
switch type
    case 'number',
    case 'name',
        decision=fis.output.mf(decision).name;
    otherwise,
        error('Допустимі значення 3 аргумента: number і name')
end
```

Для виконання *п'ятого* та *шостого завдання* необхідно здійснити навчання нечіткого класифікатора. Для розрахунку різних критеріїв навчання можна скористатися розробленою нами функцією `fclc`.

```
function delta = fclc (fis, input, crisp_target, criterion,
    mu_target, pen_value)
%FCLC - Fuzzy Classifier Learning Criteria
%Розрахунок критеріїв навчання нечіткого класифікатора.
%-------------------------------------------------------------
%fis - нечіткий класифікатор;
%input - вхідні значення вибірки даних. Один рядок відповідає
%    вхідним атрибутам одного об`єкту;
%crisp_target - вихідні значення вибірки даних (класи рішень;
%criterion - критерій навчання
%    1 - частота помилок (значення за замовченням);
%    2 - квадратична нев'язка між двома нечіткими множинами –
%        бажаними та реальними результатами класифікації;
%    3 - квадратична нев'язка між нечіткими бажаними та
%        реальними результатами класифікації з додатковим
%        штрафом за помилкове рішення;
%    4 - відстань між головними конкурентами з штрафом за
%        помилкове рішення;
%    5 - квадратична відстань між головними конкурентами з
%        штрафом за помилкове рішення.
%mu_target - бажані ступені належності до класів рішень
%    (аргумент crisp_target в нечіткій формі;
%pen_value - штрафний коефіцієнт (за замовченням - 1);
%delta - значення критерія навчання.
%-------------------------------------------------------------
%Ver. 1.0   10/12/2015
%(c)  Serhiy Shtovba and Anastasiia Galushchak,
%     Vinnytsia National Technical University
%     shtovba@ksu.vntu.edu.ua
%     www.shtovba.vinnitsa.com
%=============================================================
%Перевірка вхідних аргументів:
if (nargin<3)
    error ('Необхідно задати 3 - 6 вхідних аргументів');
end
if (nargin==3)
    criterion=1;          % < значення за замовченням
end
if (nargin==4) & ( criterion~=1)
    error('Слід задати mu_target - бажані ступені належності
        до класів рішень');
end
```

```
if (nargin<6) | (length(pen_value)==0)
    pen_value=1;          % < значення за замовченням
end
%Класифікація:
[M tmp2]=size(input);
for i=1:M
    [decision, mf_grades]=fuzzy_classifier_v(input(i,:), fis);
    D(i,1)=decision;
    MFG(i, :)=mf_grades;
end
mis_clas=(crisp_target~=D); %< визначення помилок класифікації
%Розрахунок критеріїв навчання:
switch criterion
    case 1,
        delta=mean(mis_clas);
    case 2,
        delta_mf=(MFG-mu_target).^2;
        delta=sum(sum(delta_mf')');
    case 3,
        penalty=pen_value*mis_clas+1;
        delta_mf=(MFG-mu_target).^2;
        s=sum(delta_mf')';
        delta=sum(penalty.*s);
    case 4,
        index_error=find(mis_clas==1);
        Delta_error=0;
        for i=1:length(index_error)
            j=index_error(i);
            Delta_error=Delta_error + (MFG(j, D(j))-
                MFG(j, crisp_target(j) ) )./MFG(j, D(j));
        end
        Delta_error=Delta_error*pen_value;
        index_Nerror=setdiff(1:M, index_error);
        Delta_Nerror=0;
        Z=sort(MFG(index_Nerror,:)')';
        Delta_Nerror=sum((Z(:, end)-Z(:, end-1))./Z(:, end) );
        delta=Delta_error-Delta_Nerror;
    case 5,
        index_error=find(mis_clas==1);
        Delta_error=0;
        for i=1:length(index_error)
            j=index_error(i);
            Delta_error=Delta_error+ ((MFG(j, D(j))-
                MFG(j, crisp_target(j) ) )./MFG(j, D(j)))^2;
        end
```

```
        Delta_error=Delta_error*pen_value;
        index_Nerror=setdiff(1:M, index_error);
        Delta_Nerror=0;
        Z=sort(MFG(index_Nerror,:)')';
       Delta_Nerror=sum(((Z(:,end)-Z(:,end-1))./Z(:,end)).^2);
        delta=Delta_error-Delta_Nerror;
    otherwise,  error('Недопустимий тип критерія навчання')
end
```

Детальний опис функцій `fuzzy_classifier_v` та `fclc` наведено в Довіднику ключових функцій. Навчання нечіткого класифікатора можна реалізувати за допомогою функцій умовної оптимізації, наприклад, `fmincon`. При цьому, слід встановити такі обмеження на параметри функцій належності, щоб під час навчання не порушити інтрепретуємість терм-множин. Нижче наводиться сценарій навчання нечіткого класифікатора з базою знань із 5 правил (див. табл. 10), терми якого задано гаусовою функцією належності. Налаштовуються такі параметри:

- коефіцієнти концентрацій функцій належності термів “Низький” та , “Високий” змінної x_7, термів “Низький” та “Високий” змінної x_{10}, термів “Низький”, “Середній” та “Високий” змінної x_{13};
- ядро некрайнього нечіткого терма “Середній” змінної x_{13};
- вагові коефіцієнти перших чотирьох правил (достовірність п’ятого правила не викликає сумнівів).

```
%Сценарій навчання нечіткого класифікатора для Wine Dataset
%Ресурси:  Fuzzy Logic Toolbox v.2.X
%          Optimization Toolbox v.2.X
%          FALEFC v.1.2
%Доступ до FALEFC:
%http://www.shtovba.vinnitsa.com/doc/Shtovba_book_M-files.zip
%-------------------------------------------------------------
%Ver. 2.1   21/12/2015
%(c)  Serhiy Shtovba and Anastasiia Galushchak,
%     Vinnytsia National Technical University
%     shtovba@ksu.vntu.edu.ua
%     www.shtovba.vinnitsa.com
%=============================================================
%Завантажуємо дані:
data=load('wine.data');
data=data(:, [2:end 1]);
[n,m]=size(data);
```

```
%Знаходимо номера крайніх значень:
[minData min_Index]=min(data(:, 1:m-1));
[maxData max_Index]=max(data(:, 1:m-1));
%Формуємо навчальну та тестову вибірки:
tr_index=[1:2:n min_Index max_Index];
tr_index=unique(tr_index);
usedAttr = [7 10 13];
tr_x=data(tr_index, usedAttr);
tr_y=data(tr_index, end);
test_index=setdiff(1:n, tr_index);
test_x=data(test_index, usedAttr);
test_y=data(test_index, end);
training_set=[tr_x tr_y];
test_set=[test_x test_y];
save 'wine_tr_7-10-13.dat' training_set -ASCII
save 'wine_ts_7-10-13.dat' test_set -ASCII
%Завантажуємо нечіткій класифікатор:
fis=readfis('Wine5_7-10-13.fis');
RULE_order=rule_order_fuzzy_cl(fis);
%Форматуємо навчальну вибірку для налаштування класифікатора:
[INP_tr OUT_c_tr OUT_mu_tr]=
    dp_for_fuzzy_cl_learning(fis, 'wine_tr_7-10-13.dat');
%Форматуємо тестову вибірку для перевірки класифікатора:
[INP_test OUT_c_test OUT_mu_test]=
    dp_for_fuzzy_cl_learning(fis, 'wine_ts_7-10-13.dat');
[M tmp2]=size(INP_test);
%Перевірка початкового класифікатора:
disp('Частота помилок початкового класифікатора:')
delta_tr=fclc(fis, INP_tr, OUT_c_tr , 1)
delta_test=fclc(fis, INP_test, OUT_c_test , 1)
%ПАРАМЕТРИ ДЛЯ НАЛАШТУВАННЯ
%Ваги правил:
vlb_w(1:4)=0;       %нижня межа
w_0  (1:4)=1;       %початкова точка
vub_w(1:4)=1;       %верхня межа
%Коефіцієнти функцій належності:
%        < input1 > < input2 >  <---    input3   --->
vlb_mfs=[0.7  0.7   1.5  1.5    50   50   50    500  ];
mfs_0  =[2    2     6    6      244  342  352   1071 ];
vub_mfs=[9    9     15   15     1000 1000 1000  1400 ];
%        <-------------  b  ----------------->  < c >
vlb=[vlb_mfs vlb_w];
x0= [mfs_0   w_0];
vub=[vub_mfs vub_w];
```

```
%Параметри алгоритма оптимізації:
options=optimset('Display', 'iter');
options.DiffMinChange=0.05;
options.LargeScale='off';
options.Algorithm='active-set';
options.MaxIter=15;
options.MaxFunEvals=555;
%Оптимизація:
criterion=5;
pen_val=2;
disp('Навчання:');
[xopt, delta]=fmincon(@ob_fun_wine_5rules, x0, [], [], [], [],
    vlb, vub, [], options, fis, INP_tr, OUT_c_tr, criterion,
    OUT_mu_tr, pen_val);
%Тестування після оптимізації:
tuned_fis=change_fis_wine_5rules(xopt, fis);
disp('Частота помилок після навчання:')
delta_tr=fclc(tuned_fis, INP_tr, OUT_c_tr , 1)
delta_test=fclc(tuned_fis, INP_test, OUT_c_test , 1)
fuzzy(tuned_fis)
```

Наведений вище сценарій викликає функції `ob_fun_wine_5rules` та `change_fis_wine_5rules`, лістинги яких наведено нижче.

```
function delta = ob_fun_wine_5rules(x, fis, INP, OUT_c,
    criterion, OUT_mu, pen_value)
%OB_FUN_WINE_5RULES - цільова функція задачі налаштування
%параметрів нечіткого класифікатора вин.
%-------------------------------------------------------------
%x - нові параметри класифікатора;
%fis - нечіткий класифікатор;
%INP - вхідні значення навчальної вибірки.
%OUT_c - вихідні значення навчальної вибірки;
%criterion - критерії навчання
%    1 - частота помилок (значення за замовченням);
%    2 - квадратична нев'язка між двома нечіткими множинами -
%        бажаними та реальними результатами класифікації;
%    3 - квадратична нев'язка між нечіткими бажаними і
%        реальними результатами класифікації з додатковим
%        штрафом за помилкове рішення;
%    4 - відстань між головними конкурентами з штрафом за
%        помилкове рішення;
%    5 - квадратична відстань між головними конкурентами з
%        штрафом за помилкове рішення.
```

```
%OUT_mu    - бажані ступені належності до класів рішень;
%pen_value - штрафний коефіцієнт (за замовченням - 1).
%delta - значення критерія навчання.
%-------------------------------------------------------------
%Ver. 1.3   21/12/2015
%(c)  Serhiy Shtovba and Anastasiia Galushchak,
%     Vinnytsia National Technical University
%     shtovba@ksu.vntu.edu.ua
%     www.shtovba.vinnitsa.com
%=============================================================
%Перевірка вхідних аргументів:
if (nargin<4)
    error ('Необхідно задати 4, 5, 6 або 7 аргументів');
end
if (nargin==4)
    criterion=1;    %default value
end
if (nargin==5) & ( criterion~=1)
    error ('Необхідно задати OUT_mu - вектор бажаних вихідних
            ступенів належності');
end
if (nargin<7) | (length(pen_value)==0)
    pen_value=5;     %default value
end
fis=change_fis_wine_5rules(x, fis);
delta=fclc(fis, INP, OUT_c, criterion, OUT_mu, pen_value);




function fis=change_fis_wine_5rules(x, fis)
%CHANGE_FIS_WINE_5RULES - встановлення нових параметрів x
%нечіткого класифікатора fis.
%-------------------------------------------------------------
%Ver. 1.2   19/12/2015
%(c)  Serhiy Shtovba and Anastasiya Galuchshak,
%     Vinnytsia National Technical University
%     shtovba@ksu.vntu.edu.ua
%     www.shtovba.vinnitsa.com
%=============================================================
%Нові параметри функцій належності
fis.input(1).mf(1).params(1)=x(1);
fis.input(1).mf(2).params(1)=x(2);
fis.input(2).mf(1).params(1)=x(3);
```

```
fis.input(2).mf(2).params(1)=x(4);
fis.input(3).mf(1).params(1)=x(5);
fis.input(3).mf(2).params(1)=x(6);
fis.input(3).mf(3).params(1)=x(7);
fis.input(3).mf(2).params(2)=x(8);
%Нові ваги правил:
fis.rule(1).weight=x(9);
fis.rule(2).weight=x(10);
fis.rule(3).weight=x(11);
fis.rule(4).weight=x(12);
```

Протокол навчання нечіткого класифікатора наведено нижче.

```
>>
Частота помилок початкового класифікатора:
delta_tr =  0.3300
delta_test =  0.3974

Навчання:
                                Max     Line search  Directional  First-order
 Iter F-count        f(x)   constraint   steplength   derivative   optimality Procedure
    0      13      -20.764            0
    1      26     -40.8709            0            1        -8.48         17.1
    2      39     -50.7735            0            1        -12.4         13.5
    3      52     -62.1535            0            1        -7.23         41.7
    4      65      -62.477            0            1        -1.46         8.09  Hessian modified
    5      78     -62.8232            0            1       -0.478         6.83
    6      92     -70.0876            0          0.5        -1.67          158  Hessian modified
    7     109     -70.2029            0       0.0625        -1.17          471
    8     122     -70.4986            0            1       -0.886          461
    9     136     -70.5506            0          0.5        -1.61          502
   10     163     -70.5506            0    -6.1e-005       -0.386          554
   11     190     -70.5506            0    -6.1e-005       -0.321          135
   12     206     -70.5584            0        0.125       -0.264          143
   13     233     -70.5584            0    -6.1e-005       -0.216          174
   14     260     -70.5584            0    -6.1e-005       -0.209          121
   15     287     -70.5584            0    -6.1e-005        -0.15          240

Solver stopped prematurely.

fmincon stopped because it exceeded the iteration limit,
options.MaxIter = 15 (the selected value).

Частота помилок після навчання:
delta_tr = 0.0900
delta_test =  0.1026
```

В результаті навчання частота помилок на навчальній вибірці зменшилась з 0.33 до 0.09, а на тестовій вибірці – з 0.4 до 0.1. Під час навчання змінилися функції належності (рис. 29) та ваги правил (див. табл. 10).

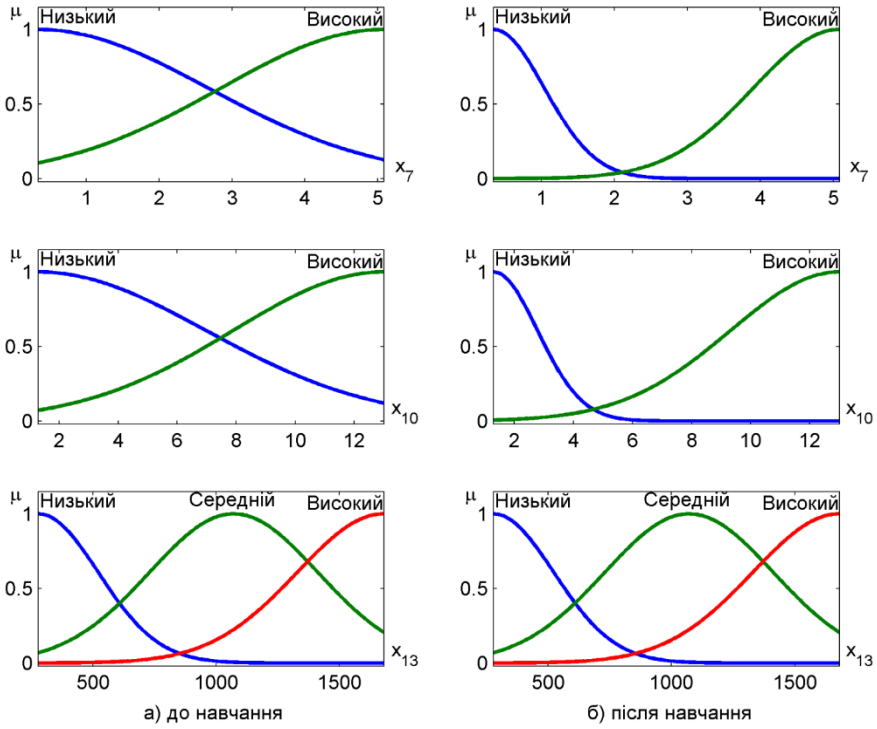


Рисунок 29 – Функції належності нечіткого класифікатора

Питання для самоконтролю

1. Сформулюйте задачу класифікації.
2. Обґрунтуйте основні переваги нечітких класифікаторів.
3. Обґрунтуйте основні недоліки нечітких класифікаторів.
4. За якими критеріями можна навчати нечіткий класифікатор?
5. Скільки має бути правил в базі знань нечіткого класифікатора?
6. Які основні способи вибору правил нечіткого класифікатора?
7. Які функції належності можна застосовувати в нечіткому класифікаторі?
8. Чому функції належності термів змінної x_{13} майже не змінилися під час навчання нечіткого класифікатора вин?
9. Які корективи потрібно внести в сценарій навчання, щоб функції належності термів змінної x_{13} налаштовувалися під час оптимізації?

Довідник ключових функцій

classregtree

Призначення Синтезує дерево рішень або регресійне дерево з вибірки даних.

Синтаксис

```
T = classregtree(X, y)
T = classregtree(X, y, Par1, val1, Par2, val2, ...)
```

Аргументи

`X` – матриця вхідних даних навчальної вибірки; пропуски даних слід позначити як `NaN`;

`y` – вектор вихідних значень (класів рішень);

`Par1`, `Par2` – назви параметрів;

`val1`, `val2` – значення параметрів.

Основні назви параметрів:

`'categorical'` – ознака категоріальних атрибутів;

`'method'` – тип задачі: `'classification'` – класифікація або `'regression'` – регресія;

`'names'` – назви атрибутів, які слід передати списком;

`'cost'` – платіжна матриця;

`'minleaf'` – мінімальна кількість об'єктів, для яких може бути сформований окремий листок дерева рішень (значення за замовченням – 1);

`'minparent'` – мінімальна кількість об'єктів на листку, за якої можливе його подальше розщеплення (значення за замовченням – 10);

`'weights'` – вектор вагових коефіцієнтів спостережень (значення за замовченням – 1);

`'splitcriterion'` – правило розщеплення: `'gdi'` – на основі індекса Джині (значення за замовченням); `'deviance'` – на основі ентропійного критерію; `'twoing'` – на основі ділення навпіл.

Вихідні змінні `T` – дерево рішень.

Приклад

```
%Створення дерева рішення для задачі класифікації з
%платіжною матрицею з урахуванням того, що атрибути
%1, 3 та 5 є категоріальними:
T = classregtree(X, y, 'method', 'classification',
   'categorical', [1 3 5], 'cost', [0 1; 1 5]);
```

eval

Призначення Класифікація за допомогою дерева рішень.

Синтаксис

```
y = eval(T, X)
y = eval(T, X, pruning_level)
[y, node_number] = eval(...)
[y, node_number, class_number] = eval(...)
```

Аргументи `T` – дерево рішень;

`X` – матриця вхідних атрибутів;

`pruning_level` – список рівнів підрізання дерева рішень.

Вихідні змінні `y` – результати класифікації об'єктів з `X` (якщо передано аргумент `pruning_level`, тоді результати класифікації розраховуються для вказаних рівнів підрізання дерева рішень);

`node_number` – результати класифікації у формі номерів вершин дерева рішень;

`class_number` – результати класифікації у формі порядкових номерів класів рішень.

Приклад

```
%Завантажуємо змінні в робочу область:
load fisheriris;
%Створюємо дерево рішень:
T = classregtree(meas, species)
view(T);
%Проводимо класифікацію:
y = eval(T, meas);
%Підраховуємо безпомилковість класифікації:
p = mean(strcmp(y, species));
```

evalfis

Призначення Нечітке логічне виведення.

Синтаксис

```
out = evalfis(inp, fis)
out = evalfis(inp, fis, numPts)
[out, IRR, ORR, ARR] = evalfis(inp, fis)
[out, IRR, ORR, ARR] = evalfis(inp, fis, , numPts)
```

Аргументи

`inp` – матриця значень вхідних змінних, для яких необхідно виконати нечіткий висновок. Розмір матриці $M \times N$, де N – кількість вхідних змінних; M – кількість вхідних даних. Кожен рядок матриці представляє один вектор значень вхідних змінних;

`fis` – система нечіткого виведення;

`numPts` – необов'язковий аргумент, що задає кількість точок дискретизації функцій належності. Значення за замовчуванням дорівнює 101, яке означає, що всі нечіткі множини представляються у вигляді 101 пари чисел "елемент універсальної множини – ступінь належності". При зменшенні точок дискретизації зростає швидкість логічного виведення і зменшується точність обчислень.

Вихідні змінні

`output` – матриця значень вихідних змінних, що отримується в результаті нечіткого висновку для даних `inp`. Розмір матриці $M \times L$, де M – кількість вхідних даних; L – кількість вихідних змінних в `fis`;

`IRR` – матриця ступенів належності вхідних значень нечітким термам з бази знань. Розмір матриці $N_{rules} \times N$, де N_{rules} – кількість правил в `fis`; N – кількість вхідних змінних;

`ORR` – матриця розміром `numPts` $\times (N_{rules} \cdot L)$. Кожен стовпець матриці містить функцію належності вихідної змінної, яку отримано в результаті виведення по одному правилу. Функція належності дискретизується на `numPts` точках і представляється вектором ступенів належності;

`ARR` – матриця розміром `numPts` $\times L$, що містить функції належності вихідних змінних – нечіткі результати логічного виведення по всім правилам. Функції належності

дискретизується на `numPts` точках і представляються вектором ступенів належності.

Змінні `IRR`, `ORR` і `ARR` є необов'язковими; вони містять проміжні результати нечіткого виведення. При завданні декількох вхідних даних значення цих змінних стосуються тільки останнього вектора даних.

Приклад

```
fis = readfis('tipper');
tip = evalfis([3 8], fis)
%Перший рядок завантажує демо-систему нечіткого
%виводу tipper, яка моделює визначення відсотка
%чайових в американському ресторані. Другий рядок
%розраховує розмір чайових, коли service = 3 та
%food = 8.
```

fclc

Призначення Розрахунок критеріїв навчання нечіткого класифікатора.

Синтаксис

```
delta=fclc(fis, input, crisp_target, criterion, mu_target, pen_value)
delta=fclc(fis, input, crisp_target, criterion, mu_target)
delta=fclc(fis, input, crisp_target, criterion)
delta=fclc(fis, input, crisp_target)
```

Аргументи

`fis` – нечіткий класифікатор;

`input` – вхідні значення вибірки даних. Один рядок відповідає вхідним атрибутам одного обєкту;

`crisp_target` – вихідні значення вибірки даних (класи рішень) – вектор-стовпчик;

`criterion` – критерії навчання: 1 – частота помилок (значення за замовченням); 2 – квадратична нев'язка між двома нечіткими множинами – бажаними та реальними результатами класифікації; 3 – квадратична нев'язка між нечіткими бажаними та реальними результатами класифікації з додатковим штрафом за помилкове рішення; 4 – відстань між головними конкурентами з штрафом за помилкове рішення; 5 – квадратична відстань між головними конкурентами з штрафом за помилкове рішення;

`mu_target` – бажані ступені належності до класів рішень

	(аргумент `crisp_target` в нечіткій формі; `pen_value` – штрафний коефіцієнт (значення за замовченням – 1).
Вихідні змінні	`delta` – значення критерію навчання.

`fuzzy_classifier_v`

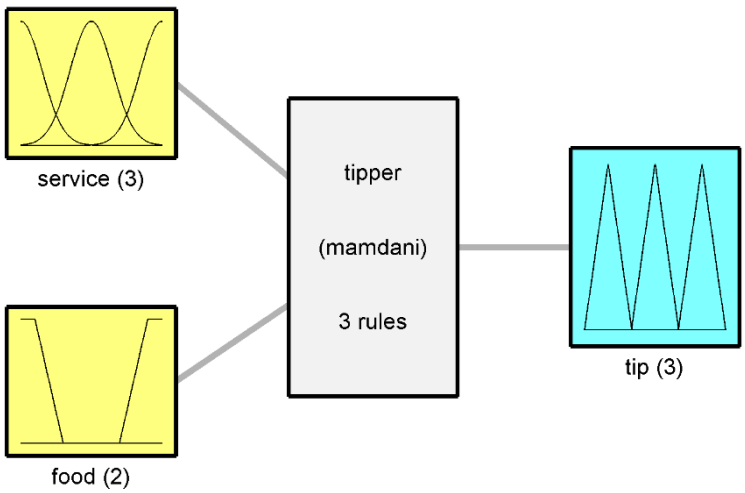
Призначення	Нечітке виведення для задачі класифікації.
Синтаксис	`[decision,mf_grades]=fuzzy_classifier_v(x, fis, type, inference)`
Аргументи	`x` – вхідний вектор (атрибути об'єкта); `fis` – структура системи нечіткого логічного виведення; `inference` – метод класифікації; можливі значення: `'single'` – рішенням обирається консеквент правила з максимальним ступеням виконання (значення за замовченням) та `'voting'` – обирається клас з максимальною сумою степенем належності, коли кожне правило голосує за свій клас. `type` – формат рішення; можливі значення: `'number'` - порядковий номер класу (значення за замовченням); `'name'` – назва класу.
Вихідні змінні	`decision` – прийняття рішення (порядковий номер або назва класу); `mf_grades` – ступені належності об'єкта до кожного класу.

`plotfis`

Призначення	Графічне представлення структури та основних параметрів системи нечіткого виведення.
Синтаксис	`plotfis (fis)`
Аргументи	`fis` – система нечіткого виведення.

Приклад

```
fis = readfis('tipper');
plotfis(fis)
%Вивід схеми нечіткої демо-системи "Tipper".
%Входи системи зображуються в лівій частині вікна,
%виходи – у правій частині, в центрі – база знань. У
%вікні виводяться найменування та тип нечіткої
системи, кількість термів і кількість правил. Також
%зображуються графіки функцій належності всіх
нечітких термів.
```



System tipper: 2 inputs, 1 outputs, 3 rules

plotmf

Призначення Виводить графіки функцій належності термів однієї змінної нечіткої системи.

Синтаксис

```
plotmf (fis, varType, varIndex)
plotmf (fis, varType, varIndex, numPts)
[X, y] = plotmf (fis, varType, varIndex)
[X, y] = plotmf (fis, varType, varIndex, numPts)
```

Аргументи

`fis` – система нечіткого виведення;

`varType` – тип змінної. Допустимі значення: `'input'` – вхідна змінна та `'output'` – вихідна змінна;

`varIndex` – порядковий номер змінної. Вхідні і вихідні змінні нумеруються незалежно;

`numPts` – кількість точок дискретизації для побудови графіків функцій належності (значення за замовчуванням – 181).

Вихідні змінні

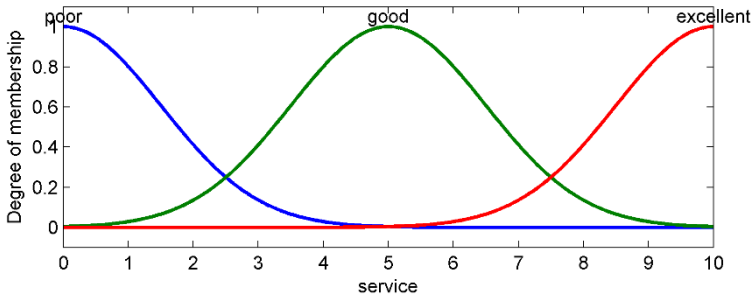
`x` – матриця абсцис-координат для всіх графіків функцій належності;

`y` – матриця значень функцій належності для аргументів `x`.

При виклику функції `plotmf` з вихідними змінними графіки функцій належності не виводяться.

Приклад

```
fis = readfis ('tipper');
plotmf (fis, 'input', 1)
%Вивід графіків функцій приналежності термів першої
%вхідної змінної демо-системи нечіткого висновку
%"Tipper".
```



prune

Призначення Підрізання дерева рішень.

Синтаксис

```
T2 = prune(T1, param, value)
```

Аргументи

`T1` – початкове дерево рішень;

`param` – назва параметра;

`value` – значення параметра.

Основні назви параметрів: `'level'` – підзрізання дерева за рівнем та `'nodes'` – підзрізання за вершинами.

Для методу `'level'` слід вказати на скільки рівнів зменшиться дерево рішень.

Для методу `'nodes'` слід вказати порядкові номери проміжних вершин, які необхідно вилучити з дерева рішень. Проміжні та термінальні вершини дерева пронумеровано порівнево. Номери проміжних вершин можна визначити з графічного вікна, яке створюється за допомогою функції view. Список проміжних вершин дерева `T1` можна отримати таким чином: `unique(T1.parent(2:end))`.

Вихідні змінні `T2` – підрізане дерево рішень.

Приклад
```
%Підрізання дерева рішень на 2 рівня:
T2 = prune(T1, 'level', 2);
view(T2)
```

readfis

Призначення Завантаження з файлу системи нечіткого виведення.

Синтаксис
```
fis = readfis
fis = readfis('fisfile')
```

Аргументи `'fisfile'` – назва файлу, в якому зберігається система нечіткого виведення. Виклик функції без вхідного аргумента призводить до появи типового вікна відкриття файлу.

Вихідні змінні `fis` – структура системи нечіткого виведення.

Приклад
```
%Завантаження в робочу область нечіткої
%демо-системи "Tipper":
fis = readfis ('tipper')
```

showrule

Призначення Вивід бази правил системи нечіткого виведення.

Синтаксис
```
showrule(fis)
showrule(fis, ruleIndex, ruleFormat, lang)
outStr = showrule(fis)
outStr = showrule(fis, ruleIndex, ruleFormat, lang)
```

Аргументи `fis` – система нечіткого виведення;

`ruleIndex` – порядкові номери правил, які будуть показані (за замовчуванням виводяться всі правила);

`ruleFormat` – формат виведення правил. Допустимі значення: `'verbose'` – словесний (встановлено за замовчанням), `'Symbolic'` – символьний та `'Indexed'` – індексний;

`lang` – мова правил у форматі `'verbose'`. Допустимі значення: `'english'` – англійська (за замовчанням); `'francais'` – французька; `'deutsch'` – німецька.

Вихідні змінні `outStr` – список правил бази знань системи нечіткого виведення `fis`.

Приклад

```
%Виведення на екран бази знань нечіткої
%демо-системи "Tipper":
fis = readfis ('tipper');
showrule (fis)
```

1. If (service is poor) or (food is rancid) then (tip is cheap) (1)
2. If (service is good) then (tip is average) (1)
3. If (service is excellent) or (food is delicious) then (tip is generous) (1)

test

Призначення Розрахунок показників точності дерева рішень.

Синтаксис

```
MCR = test(T, mode, X, y)
[MCR, Delta_MCR, Num_nodes, Best_level] = test(...)
[...] = test(..., par1, val1, par2, val2, ...)
```

Аргументи

`T` – дерево рішень;

`X` – матриця вхідних значень вибірки даних;

`y` – вектор вихідних значень вибірки даних;

`mode` – режим перевірки дерева рішень.

`par1`, `par2` – назви параметрів;

`val1`, `val2` – значення параметрів.

Режими перевірки дерева рішень:

`mode='r'` (скорочення від resubstitution) – перевірка на навчальній вибірці, яка зберігається в структурі `T`. В цьому випадку аргументи `X` та `y` передавати не потрібно;

`mode='t'` (скорочення від test) – перевірка на тестовій вибірці, яку потрібно передати аргументами `X` та `y`;

`mode='c'` (скорочення від cross-validate) – перехресна перевірка (крос-валідація) на навчальній вибірці, яку потрібно передати аргументами `X` та `y`. Випадковим чином формуються підвибірки приблизно однакового розміру, потім дерево рішень навчають на кожній з підвибірок з перевіркою на решті даних та осереднюють результати тестування.

Основні назви параметрів:

`'weights'` – ваги об’єктів в вибірці;

`'nsamples'` – кількість підвибірок для перехресної перевірки (значення за замовченням – 10);

`'treesize'` – критерій оптимальності. Можливі значення:

'se' – дерево з мінімальною помилкою з урахуванням довірчого інтервалу її визначення (значення за замовченням – 10) та 'min' – дерево з мінімальною помилкою без урахування довірчого інтервалу.

Вихідні змінні

MCR – безпомилковість (або ризик, якщо відома платіжна матриця) дерев рішень, підрізаних на різних рівнях;

Delta_MCR – оцінка точності визначення MCR за (3);

Num_nodes – кількість листків, підрізаних на різних рівнях;

Best_level – оптимальний рівень підрізання дерева рішень.

Приклад

```
%Завантажуємо змінні в робочу область:
load fisheriris;
%Створюємо дерево рішень:
T = classregtree(meas, species)
view(T);
%Проводимо класифікацію:
y = eval(T, meas);
%Знаходимо оптимальне дерево за допомогою
%перехресної перевірки:
[MCR, D, node, best] = test(T, 'c', meas, species);
Tbest = prune(T, 'level', best);
```

view

Призначення Створення графічного зображення дерева рішень.

Синтаксис

```
view(T)
view(T, par1, val1, par2, val2)
```

Аргументи

T – дерево рішень;

par1, par2 – назви параметрів;

val1, val2 – значення параметрів.

Допустимі назви параметрів:

'names' – назви вхідних змінних згідно до їх порядку в навчальній вибірці;

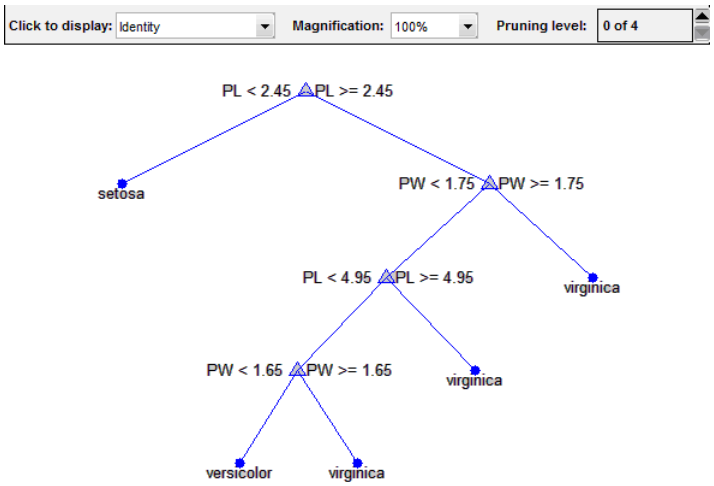
'prunelevel' – кількість рівнів підрізання дерева.

Значення параметру 'names' задається списком, наприклад, {'x_1' 'x_2' 'x_3' 'x_4'}.

Приклад

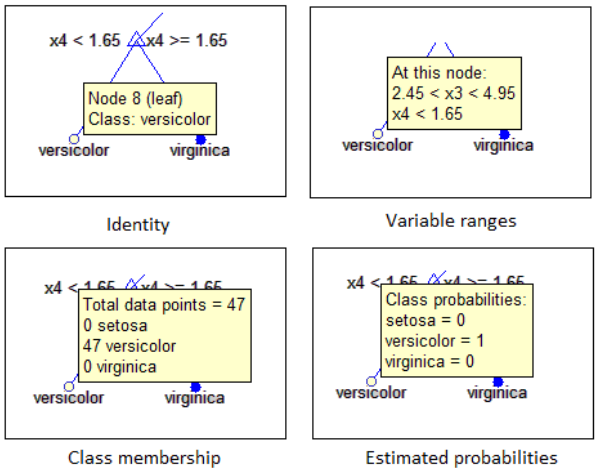
```
load fisheriris;  %завантажуємо дані про іриси
%Створюємо дерево рішень:
T = classregtree(meas, species,...
                 'names',{'SL' 'SW' 'PL' 'PW'})
view(T);
```

В результаті отримуємо наступне графічне вікно



Вікно має лічильник Pruning Level за допомогою якого можна інтерактивно підрізати дерево рішень. Дії у цьому вікні не змінюють дерево рішень з робочої області.

При виділені вершини курсором миші виводиться інформація згідно до обраного режиму з списку Click to display. Можливі такі режими: Identity – порядковий номер вершини та відповідне правило чи клас; Variable range – межі, що їх можуть приймати атрибути в вершині; Class membership – кількість об’єктів кожного класів, що потрапили у вершину; Estimated probability – ймовірність появи кожного класу в вершині.



Література

1. Андреев И. М. Описание алгоритма CART / И.М. Андреев // Exponenta Pro: Математика в приложениях. – 2004. – №3–4. – С. 48–53.
2. Воронцов К. В. Комбинаторный подход к оценке качества обучаемых алгоритмов / К. В. Воронцов // Математические вопросы кибернетики. – 2004. – Т. 13. – С. 5–36.
3. Городецкий В. И. Методы и алгоритмы коллективного распознавания / В.И. Городецкий, С.В. Серебряков // Автоматика и телемеханика. – 2008. – №11. – С. 3-40.
4. Дьяконов А. Г. Анализ данных, обучение по прецедентам, логические игры, системы WEKA, RapidMiner и MatLab (практикум на ЭВМ кафедры математических методов прогнозирования) / А.Г. Дьяконов. – МАКСПресс, 2010. – 278 с.
5. Ивахненко А. Г. Долгосрочное прогнозирование и управление сложными системами / А. Г. Ивахненко. –К.: Техніка, 1975. – 312 с.
6. Лю Б. Теория и практика неопределенного программирования / Б. Лю. – М.: БИНОМ. Лаборатория знаний, 2005. – 416 с.
7. Начало работы с MATLAB / Пер. с англ. Конюшенко В. В. – Softline Co., 2014. – 74 с. Режим доступа: http://www.exponenta.ru/educat/free/matlab/gs.pdf .
8. Орлов А. И. Прикладная статистика. Учебник. / А. И. Орлов. – М.: Издательство “Экзамен”, 2004. – 656 с.
9. Растригин Л.А. Адаптация сложных систем. Методы и приложения / Л. А. Растригин. – Рига: Зинатне, 1981. – 375 с.
10. Ротштейн А. П. Интеллектуальные технологии идентификации / А. П. Ротштейн. – Винница: Вінниця–УНІВЕРСУМ, 1999. – 320 с.
11. Ротштейн О. П. Проектування нечітких баз знань: лабораторний практикум та курсове проектування: навч. посіб. / О. П. Ротштейн , С. Д. Штовба. – Вінниця: Вінницький державний технічний університет, 1999. – 65 с.

12. Шахиди А. Деревья решений / А. Шахиди – BaseGroup Labs, 2014. – Режим доступа: http://www.basegroup.ru/library/analysis/tree/

13. Штовба С. Д. Анализ критериев обучения нечеткого классификатора / С. Д. Штовба, О. Д. Панкевич, А. В. Нагорна // Автоматика и вычислительная техника. – 2015. – №3.– С. 5–16.

14. Штовба С. Д. Вплив кількості нечітких правил на точність бази знань Мамдані / С. Д. Штовба, В. В. Мазуренко, О. Д. Панкевич // Вісник Хмельницького національного університету. Технічні науки. – 2011. – №2. – С. 185–188.

15. Штовба С. Д. Генетичний алгоритм вибору правил нечіткої бази знань, збалансованої за критеріями точності та компактності / С. Д. Штовба, В. В. Мазуренко, Д. А. Савчук // Наукові праці Вінницького національного технічного університету. – 2012. – №3. – Режим доступу: http://praci.vntu.edu.ua/index.php/praci/article/view/331 .

16. Штовба С. Д. Дослідження навчання компактних нечітких баз знань типу Мамдані / С. Д. Штовба, В. В. Мазуренко // Штучний інтелект. – 2011. – №4 – С. 521–529.

17. Штовба С. Д. Інтелектуальні технології ідентифікації залежностей. Лабораторний практикум / С. Д. Штовба, В. В. Мазуренко. Ел. навч. посіб. – Вінниця: Вінницький національний технічний університет, 2014. – 113 с. Режим доступу: http://ir.lib.vntu.edu.ua/handle/123456789/2294 .

18. Штовба С. Д. Критерії точності та компактності для оцінювання якості нечітких баз знань в задачах ідентифікації / С. Д. Штовба, О. В. Штовба , О. Д. Панкевич // Наукові праці Вінницького національного технічного університету. – 2012. – №4. – Режим доступу: http://praci.vntu.edu.ua/index.php/praci/article/view/343 .

19. Штовба С. Д. Обеспечение точности и прозрачности нечеткой модели Мамдани при обучении по экспериментальным данным / С. Д. Штовба // Проблемы управления и информатики. – 2007. – №4. – С. 102–114.

20. Штовба С. Д. Порівняння критеріїв навчання нечіткого класифікатора / С. Д. Штовба // Вісник Вінницького політехнічного інституту. – 2007. – №6. – С. 84–91.

21. Штовба С. Д. Проектирование нечетких систем средствами MATLAB / С. Д. Штовба. – М.: Горячая линия – Телеком, 2007. – 288 с.

22. Bache K. UCI Machine Learning Repository / K. Bache, M. Lichman. Irvine: University of California, School of Information and Computer Science. 2014. – Режим доступу: http://archive.ics.uci.edu/ml .

23. Ishibuchi H. Classification and modeling with linguistic information granules: advanced approaches advanced approaches to linguistic data mining / H. Ishibuchi, T. Nakashima, M. Nii – Berlin – Heidelberg: Springer-Verlag, 2005. – 307 p.

24. Kuncheva L. Combining pattern classifiers: methods and algorithms / L. Kuncheva. – John Wiley & Sons, 2004. – 350 p.

25. Statistics Toolbox User's Guide. – Mathworks Inc., 2014. – Режим доступу: http://www.mathworks.com/help/pdf_doc/stats/stats.pdf .

Навчальне видання

Штовба Сергій Дмитрович
Галущак Анастасія Володимирівна

ІДЕНТИФІКАЦІЯ БАГАТОФАКТОРНИХ ЗАЛЕЖНОСТЕЙ ЗА ДОПОМОГОЮ БАЗ ЗНАНЬ

Лабораторний практикум

Оригінал-макет підготовлено авторами.
Друкується в авторській редакції.

Підписано до друку 22.12.2015
Формат 29,7×42¼
Гарнітура Times New Roman.

Вінницький національний технічний університет,
навчально-методичний відділ ВНТУ.
21021, м. Вінниця, Хмельницьке шосе, 95.
ВНТУ, к. 2201.
Тел. (0432) 59-87-36.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.

publish.vntu.edu.ua; email: kivc.vntu@gmail.com.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.

Штовба Сергій Дмитрович

Професор, доктор технічних наук, професор кафедри комп`ютерних систем управління Вінницького національного технічного університету.

В 1993 р. закінчив Вінницький політехнічний інститут за спеціальністю «Конструювання та технологія електронно-обчислювальних засобів».

В 1997 р. захистив кандидатську дисертацію з математичного моделювання, а в 2009 р. – докторську дисертацію з інформаційних технологій.

Автор 5 книг та близько 100 статей з теорії та застосування обчислювального інтелекту.

www.shtovba.vk.vntu.edu.ua
shtovba@ksu.vntu.edu.ua
www.shtovba.vinnitsa.com
shtovba@gmail.com

Галущак Анастасія Володимирівна

Асистент кафедри комп`ютерних систем управління Вінницького національного технічного університету.

В 2012 р. закінчила Вінницький національний технічний університет за спеціальністю «Екологія та охорона навколишнього середовища».

Автор понад 10 наукових праць.

a.v.galushchak@vntu.edu.ua

IEBN 804-479-000007-31