С.Д. Штовба
В.В. Мазуренко

Інтелектуальні технології ідентифікації залежностей

ЛАБОРАТОРНИЙ ПРАКТИКУМ

Міністерство освіти і науки України

Вінницький національний технічний університет

С. Д. Штовба

В. В. Мазуренко

Інтелектуальні технології ідентифікації залежностей. Лабораторний практикум

навчальний посібник

Вінниця

ВНТУ

2014

УДК 681.3.06(075)
ББК 22.18я73
Ш92

Рекомендовано до друку Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол №10 від 26.06.2014 р.).

Рецензенти:
О. В. Бісікало, доктор технічних наук, доцент
А. Я. Кулик, доктор технічних наук, професор
Д. І. Катєльніков, кандидат технічних наук, доцент

Штовба С.Д.
Ш92 Інтелектуальні технології ідентифікації залежностей. Лабораторний практикум : електронний навчальний посібник / Штовба С.Д., Мазуренко В.В. – Вінниця : ВНТУ, 2014. – 113 с.

У навчальному посібнику наведено теоретичний матеріал з технологій ідентифікації багатофакторних залежностей за допомогою регресійного аналізу, дерев рішень, нечітких баз знань та нейронних мереж. Навчальний посібник призначено для студентів напряму підготовки 6.050202 “Автоматизація та комп'ютерно-інтегровані технології”, що вивчають дисципліни “Інтелектуальні технології” та “Інтелектуальні системи в менеджменті і бізнесі”. Він може застосовуватися під час викладення дисципліни “Сучасні інформаційні технології в науці та освіті” магістрантам усіх спеціальностей.

УДК [659.1+339.138] 681.3.06 (075)

ББК 22.18

IEBN 804-479-000001-14

ЗМІСТ

Передмова

В техніці, економіці, медицині, біології, політиці, спорті та в інших областях накопичені результати спостережень за різноманітними багатофакторними залежностями. Задача ідентифікації полягає в побудові за результатами спостережень математичних моделей аналізованих багатофакторних залежностей. Ідентифікацію залежностей 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 описувала експериментальні дані. Цей етап найбільш формалізований і, зазвичай, зводиться до вирішення задачі оптимізації із мінімізації середньої квадратичної нев'язки для об'єктів з неперервним виходом, чи рівня безпомилкості класифікації для об'єктів з дискретним виходом. Масштабність реальних задач ідентифікації, наявність в результатах спостережень нечислових даних та пропусків обумовлює залучення для їх вирішення інтелектуальних технологій, які добре пристосовані для опрацювання невизначеності та прийняття раціональних рішень в умовах великої обчислювальної складності конкурентних точних алгоритмів.

Незважаючи на велику кількість друкованих праць та інтернет-ресурсів з теорії ідентифікації сьогодні недостатньо систематизовано наукові знання, які були б корисними під час вирішення реальних задач з використанням сучасних інформаційних технологій. Більшість книжок з теорії ідентифікації акцентовано на строгості математичних викладок. При цьому в них майже не згадуються сучасні математичні пакети, в яких автоматизовано більшість алгоритмів ідентифікації. В результаті вивчення ідентифікації за такими книжками студент може отримати добрі теоретичні знання і навіть на їх основі зможе вирішити прості (іграшкові) задачі. Але буде майже безпорадним у разі потреби вирішення практичних задач ідентифікації у стислі терміни – бо він не тільки не володіє сучасним

інструментарієм, але не знає ні про його можливості, ні про особливості реальних задач. Спроби ж самостійного вивчення сучасних математичних пакетів у більшості випадків закінчуються отриманням навиків з автоматизації знаходження інтегралів, матричних обчислень чи побудови трьохвимірних графіків функцій.

Під час написання посібника автори намагалися не тільки викласти теоретичний матеріал, але і показати як вирішувати практичні задачі ідентифікації з допомогою сучасного інструментарію – програмної системи MATLAB 7. Посібник складається з чотирьох лабораторних робіт, кожна з яких являє невелике закінчене наукове дослідження. Ще років 15 назад кожна з лабораторних робіт за трудомісткістю відповідала принаймні курсовій роботі, але сьогодні багато математичної рутини автоматизовано. Особливістю перших двох лабораторних робіт є проведення дослідження на реальних задачах з репозиторію з машинного навчання Каліфорнійського університету в Ірвіні. Ці задачі стали де-факто міжнародним стандартом для тестування нових методів ідентифікації. В зв'язку з цим студентам необхідно порівняти власні результати ідентифікації залежності з результатами, які опублікували конкуренти в своїх наукових статтях.

Посібник складають 4 лабораторні роботи, які стосуються різних технологій ідентифікації залежностей: регресійного аналізу, дерев рішень, нечітких баз знань та нейронних мереж. Хоча перша лабораторна робота з регресійного аналізу не стосується інтелектуальних технологій, але вона включена до посібнику як така, що дає базові знання з статистичної обробки даних в математичних пакетах. Передбачається, що студенти володіють ключовими навиками роботи в програмному середовищі MATLAB. Для новачків рекомендуємо компактний посібник [11]. При підготовці посібника використано таку літературу: для лабораторної роботи №1 – [6, 7, 11, 28]; для лабораторної роботи №2 – [1–4, 12, 25, 28]; для лабораторної роботи №3 – [13–15, 18–22]; для лабораторної роботи №4 – [5, 8, 9, 13, 15, 16, 24, 26, 27]. Цю літературу рекомендуємо для поглибленого вивчення матеріалу. Уся література доступна в інтернеті.

По виконанню кожної лабораторної роботи необхідно оформити звіт, в якому окрім постановки задачі та результатів виконання завдань необхідно навести лістинги усіх програм та основні графічні вікна. В звіті особливу увагу слід приділити висновкам та інтерпретації результатів.

1. Визначення оптимальної регресійної моделі

Мета – віднайти регресійну модель, яка найкращим чином опише залежність в експериментальних даних.

Теоретичні відомості

Розглядається залежність з n входами $X = (x_1, x_2, ..., x_n)$ та одним виходом y. Вважається відомою лише вибірка експериментальних даних про цю залежність:

$$(X_j, y_j), \quad j = \overline{1, M}, \tag{1}$$

де M – довжина вибірки;

$X_j = (x_{j1}, x_{j2}, ..., x_{jn})$.

У вибірці (1) вхідні та вихідна змінні приймають числові значення.

Необхідно віднайти теоретичну залежність $y = f(X)$, яка найкраще опише експериментальні дані. Критерієм оптимальності є середня квадратична нев'язка між експериментальними та теоретичними результатами. Середня квадратична нев'язка моделі $y = f(X)$ на вибірці (1) розраховується таким чином:

$$RMSE = \sqrt{\frac{1}{M} \sum_{j=\overline{1, M}} \left(y_j - f(X_j)\right)^2}. \tag{2}$$

В модель $y = f(X)$ входять коефіцієнти, значення яких підбирають таким чином, щоб мінімізувати нев'язку (2). Така процедура знаходження коефіцієнтів моделі називається регресійним аналізом. Відповідно,

залежність $y = f(X)$ називається регресією. Регресійний аналіз називається лінійним, якщо залежність $y = f(X)$ є лінійною:

$$y = a_0 + \sum_{i=1,\,n} a_i x_i \,, \tag{3}$$

де $a_0, a_1, ..., a_n$ – коефіцієнти.

Приклад лінійного регресійного аналізу залежності тривалості розгону автомобіля до швидкості 60 миль за год. (y) від потужності двигуна (x) наведено на рис. 1. Регресійний аналіз здійснено за даними задачі Auto MPG з репозиторію [23].

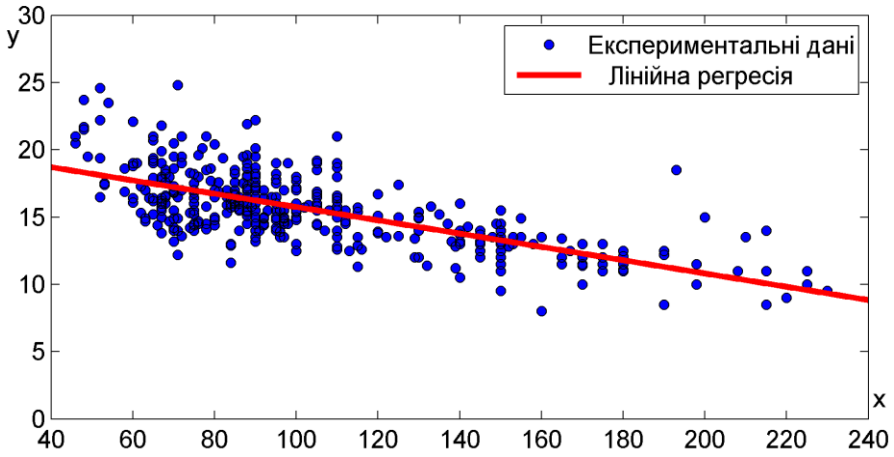


Рисунок 1 – Приклад лінійного регресійного аналізу

Для знаходження коефіцієнтів лінійної регресії необхідно вирішити таку систему рівнянь:

$$\begin{cases} \dfrac{\partial RMSE}{\partial a_0} = 0 \\ \dfrac{\partial RMSE}{\partial a_1} = 0 \\ \vdots \\ \dfrac{\partial RMSE}{\partial a_n} = 0 \end{cases}.$$

Для найпростішого випадку – для одновимірного лінійного регресійного аналізу коефіцієнти a_0 та a_1 залежності $y = a_0 + a_1 x$ розраховуються таким чином:

$$a_1 = \frac{\overline{xy} - \bar{x} \cdot \bar{y}}{\overline{x^2} - \bar{x} \cdot \bar{x}},$$

$$a_0 = \bar{y} - a_1 \bar{x},$$

де $\bar{x} = \frac{1}{M} \sum_{j=\overline{1, M}} x_j$;

$\bar{y} = \frac{1}{M} \sum_{j=\overline{1, M}} y_j$;

$\overline{x^2} = \frac{1}{M} \sum_{j=\overline{1, M}} x_j^2$;

$\overline{xy} = \frac{1}{M} \sum_{j=\overline{1, M}} x_j \cdot y_j$.

Аналітично можна знайти коефіцієнти деяких інших регресійних залежностей, але відповідні формули будуть громіздкими. Компактний запис можна отримати за матричного представлення регресійного рівняння. Розглянемо регресійне рівняння $y = a_0 + a_1 x_1 + a_2 x_2 + \ldots + a_n x_n$. Нехай поточні значення вхідних змінних є такими: $\left(x_1^*, x_2^*, \ldots, x_n^*\right)$. Тоді розрахунок вихідного значення можна реалізувати у такій матричній формі:

$$y = \left(1, x_1^*, x_2^*, \ldots, x_n^*\right) \times \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}. \qquad (4)$$

Підставляючи в (4) навчальну вибірку (1), отримуємо:

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1n} \\ 1 & x_{21} & x_{22} & \ldots & x_{2n} \\ \vdots & & & & \\ 1 & x_{M1} & x_{M2} & \ldots & x_{Mn} \end{pmatrix} \times \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}. \qquad (5)$$

Вирішуючи рівняння (5), знаходимо невідомі регресійні коефіцієнти:

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} \\ 1 & x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & & & & \\ 1 & x_{M1} & x_{M2} & \dots & x_{M2} \end{pmatrix} \backslash \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{pmatrix}. \qquad (6)$$

По аналогії з (6) можна в матричній формі описати процедуру знаходження коефіцієнтів для багатьох популярних нелінійних регресійних моделей. Наприклад, для моделі $y = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1^2 + a_4 x_1 x_2$, коефіцієнти знаходяться таким чином:

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & x_{12} & x_{11}^2 & x_{11}x_{12} \\ 1 & x_{21} & x_{22} & x_{21}^2 & x_{21}x_{22} \\ \vdots & & & & \\ 1 & x_{M1} & x_{M2} & x_{M1}^2 & x_{M1}x_{M2} \end{pmatrix} \backslash \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{pmatrix}.$$

Для складних регресійних моделей коефіцієнти знаходять за допомогою пошукових методів оптимізації.

Регресійні коефіцієнти визначають за усіма експериментальними даними тільки у тому випадку, коли наперед достовірно відомо структуру моделі. Іншими словами, коли з деяких причин відомо, що залежність є лінійною, чи квадратичною, чи логарифмічною тощо.

Якщо структура залежності наперед невідома, то для її визначення застосуємо принцип зовнішнього доповнення. Для цього експериментальні дані розіб'ємо на 2 вибірки – навчальну та тестову. Далі згенеруємо множину моделей з різною структурою – лінійні, квадратичні, кубічні тощо. За навчальною вибіркою для кожної моделі-кандидату визначимо оптимальні значення коефіцієнтів. Після цього кожну модель перевіримо на тестовій вибірці. Найкращою оберемо модель, для якої критерій (2) на тестовій вибірці є мінімальним.

Описаний підхід запобігає переускладненню моделі. На перший погляд здається, що чим складніша модель, тим вона точніша. Тобто,

кубічна модель, буде точніша за квадратичну, а квадратична – за лінійну. Якщо побудувати залежність нев'язки (2) на навчальній вибірці від складності моделі, наприклад, від загальної кількості регресійних коефіцієнтів, то вона буде монотонно спаданою (рис. 2). Коли кількість регресійних коефіцієнтів моделі дорівнюватиме M нев'язка на навчальній вибірці буде мінімальною. Але якщо перевірити отримані моделі на тестовій вибірці, то зазвичай виявляється, що зі збільшенням складності моделі нев'язка спочатку зменшується, а потім починає зростати. На рис. 2 оптимальною є регресійна модель з чотирма коефіцієнтами. Подальше ускладнення моделі призводить до втрати генералізуючих властивостей. Іншими словами, замість того, щоб відновити тенденцію в даних, модель починає описувати шум.

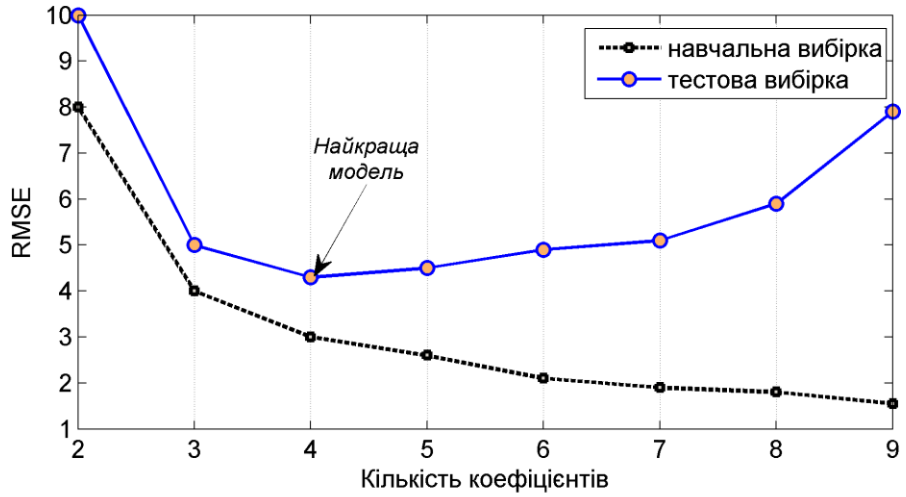


Рисунок 2 – Типові залежності нев'язки від складності моделі

Завдання на лабораторну роботу

В лабораторній роботі відновлюється двофакторна регресійна залежність $y = f(x_1, x_2)$ для однієї із задач репозиторію автоматичного навчання Каліфорнійського університету в Ірвіні [23]. Лабораторна робота полягає у виконанні таких завдань.

1. Сформулювати змістовну постановку задачі згідно варіанту з табл. 1.

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

3. Зобразити експериментальні дані у формі однофакторних залежностей.

4. Віднайти коефіцієнти шести регресійних залежностей, серед яких 2 моделі слід згенерувати самостійно. Обов'язковими є такі 4 моделі:

$$y = a_0 + a_1 x_1 + a_2 x_2;$$

$$y = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2;$$

$$y = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1^2 + a_4 x_2^2;$$

$$y = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1^2 + a_4 x_2^2 + a_5 x_1 x_2.$$

5. Обрати оптимальну регресійну модель та обґрунтувати свій вибір. Візуалізувати оптимальну регресійну модель разом з експериментальними даними.

6. Порівняти власні дослідження з конкурентними результатами інших науковців із вирішення цієї же задачі.

Таблиця 1 – Варіанти завдання

Варіант	Задача	Фактори впливу
1	Auto MPG	horsepower acceleration
2	Auto MPG	displacement year
3	Auto MPG	displacement horsepower
4	Auto MPG	horsepower year
5	Auto MPG	weight acceleration
6	Housing	NOX LSTAT
7	Housing	RM PTRATIO
8	Housing	PTRATIO LSTAT
9	Housing	B LSTAT

Варіант	Задача	Фактори впливу
10	Wine Quality (Red)	pH alcohol
11	Wine Quality (Red)	fixed acidity alcohol
12	Wine Quality (White)	sulphates alcohol
13	Wine Quality (White)	pH alcohol
14	Yacht Hydrodynamics	Length-displacement ratio Froude number
15	Yacht Hydrodynamics	Froude number Longitudinal position of the center of buoyancy
16	SkillCraft1 Master Table	LeagueIndex ComplexUnitsMade
17	SkillCraft1 Master Table	ComplexUnitsMade Age
18	SkillCraft1 Master Table	HoursPerWeek MinimapRightClicks
19	SkillCraft1 Master Table	APM: Action per minute SelectByHotkeys
20	Physicochemical Properties of Protein Tertiary Structure	Total surface area Fractional area of exposed non polar residue
21	Physicochemical Properties of Protein Tertiary Structure	Total surface area Fractional area of exposed non polar part of residue
22	Physicochemical Properties of Protein Tertiary Structure	Non polar exposed area Secondary structure penalty
23	Physicochemical Properties of Protein Tertiary Structure	Fractional area of exposed non polar residue Secondary structure penalty.

Варіант	Задача	Фактори впливу
24	Physicochemical Properties of Protein Tertiary Structure	Fractional area of exposed non polar part of residue Secondary structure penalty

Приклад виконання роботи

1. Розглядається задача Auto MPG про прогнозування паливної ефективності легкового автомобіля. Опис задачі наведено за адресою http://archive.ics.uci.edu/ml/datasets/Auto+MPG. Паливна ефективність (y) визначається кількістю миль, які можна проїхати на одному галоні палива (MPG – miles per gallon). База даних Auto MPG сформовано за 392 моделями автомобілів. Кожна модель автомобіля описана такими атрибутами: кількість циліндрів, об'єм двигуна, потужність двигуна, маса автомобіля, час розгону до швидкості 60 миль на год., рік випуску моделі, виробник та назва моделі. Дослідження проведемо для залежності паливної ефективності (y) від маси автомобіля (x_1) та року випуску (x_2).

2. Експериментальні дані розіб'ємо на навчальну та тестову вибірки. В навчальну включимо непарні рядки даних, а в тестову – парні. Для перевірки репрезентативності вибірок розрахуємо математичне сподівання (mean) та дисперсію (std) даних за кожною змінною. Результати перевірки (табл. 2) вказують на приблизно однакові значення цих показників для навчальної та тестової вибірок, тому вважаємо їх репрезентативними. Візуально підтверджує цей висновок рис. 3.

3. Однофакторні залежності (рис. 4) вказують на сильну тенденцію зменшення паливної ефективності зі збільшенням маси автомобіля та слабку тенденцію зростання її для більш нових моделей.

Таблиця 2 – Математичне сподівання та дисперсія даних

Вибірка	mean(x_1)	mean(x_2)	mean(y)	std(x_1)	std(x_2)	std(y)
Навчальна	2989.6	75.97	23.42	828.2	3.67	7.88
Тестова	2965.6	75.98	23.47	872	3.69	7.75

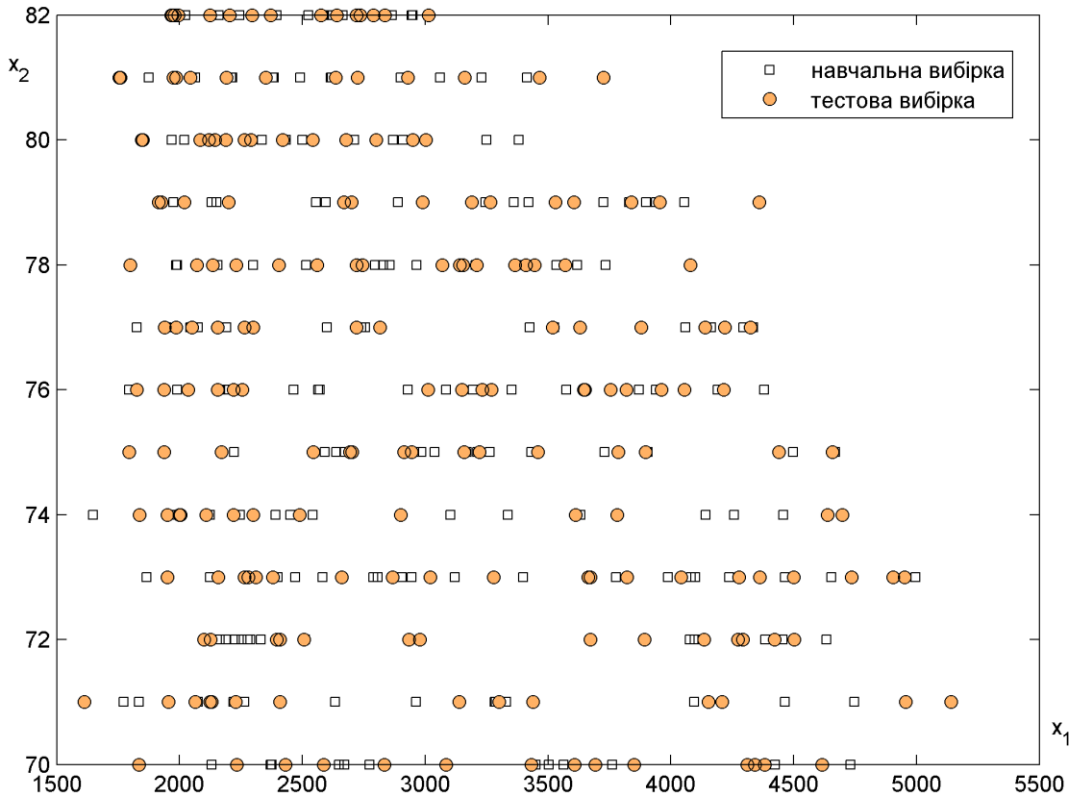


Рисунок 3 – Розподіл даних в навчальній та тестовій вибірках

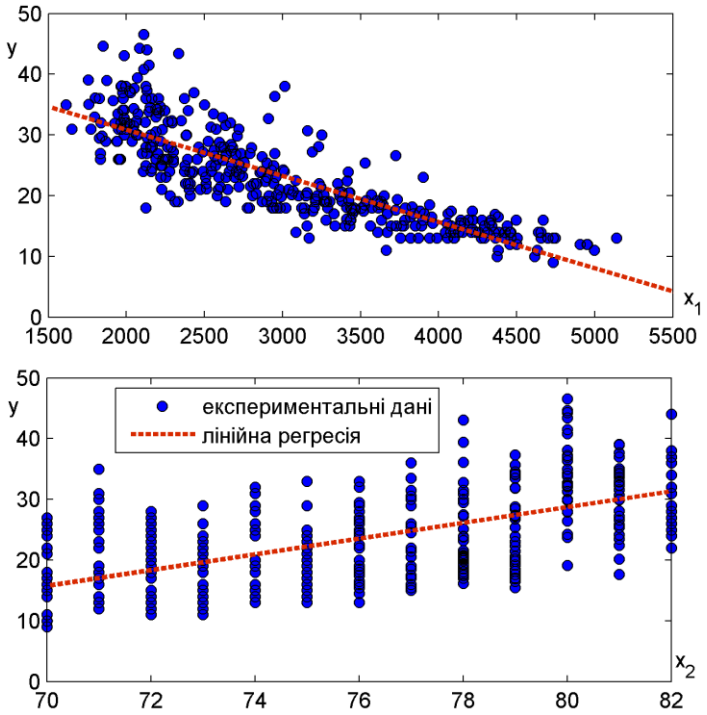


Рисунок 4 – Однофакторні залежності

4. Знайдемо коефіцієнти лінійної регресії за допомогою такої програми:

```
%-------------------------------------------------------------
%Двофакторний лінійний регресійний аналіз для задачі Auto MPG.
%-------------------------------------------------------------
%Зчитування даних:
[data, input_name] = loadgas;
vars=[4 6 7];
%Формування навчальної та тестової вибірок:
tr_set = data(1:2:end, vars);
test_set = data(2:2:end, vars);
%Довжина навчальної вибірки:
M_tr=length(tr_set);
disp('Лінійний регресійний аналіз:')
X_tr=[ones(M_tr,1) tr_set(:,1) tr_set(:,2)];
Y_tr=tr_set(:,3);
A=X_tr\Y_tr
%Розрахунок нев'язки на навчальній вибірці:
pred_tr=X_tr*A;
RMSE_tr=norm(Y_tr-pred_tr)/sqrt(M_tr)
%Розрахунок нев'язки на тестовій вибірці:
M_test=length(test_set);
X_test=[ones(M_test,1) test_set(:,1) test_set(:,2)];
Y_test=test_set(:,3);
pred_test=X_test*A;
RMSE_test=norm(Y_test-pred_test)/sqrt(M_test)
%Побудова графіків:
subplot(2 ,3,1)
plot(Y_tr, pred_tr, 'ko', 'markersize', 5)
hold on
plot([5 50],[5 50], 'r-', 'linewidth', 1)
xlabel('Експеримент')
ylabel('Теорія')
RMSE_str=num2str(RMSE_tr, 3);
title(['Лінійна регресія, RMSE_t_r=', RMSE_str])
subplot(2,3,2)
plot(Y_test, pred_test, 'ko', 'markersize', 5)
hold on
plot([5 50],[5 50], 'r-', 'linewidth', 1)
xlabel('Експеримент')
ylabel('Теорія')
RMSE_str=num2str(RMSE_test, 3);
title(['Лінійна регресія, RMSE_t_e_s_t=', RMSE_str])
```

Після незначної модифікації програми знайдемо ще 5 інших регресійних моделей. Усі моделі зведемо в табл. 3.

Таблиця 3 – Результати регресійного аналізу

№	Модель	$RMSE_{tr}$	$RMSE_{test}$
1	$y = -15.37 - 0.007x_1 + 0.779x_2$	3.5	3.34
2	$y = -129.05 + 0.034x_1 + 2.296x_2 + 0.0005x_1x_2$	3.2	3.19
3	$y = 336.51 - 0.023x_1 - 7.96x_2 + 0x_1^2 + 0.058x_2^2$	3	2.88
4	$y = 166.7 - 0.001x_1 - 4.258x_2 + 0x_1^2 + 0.0382x_2^2 - 0.0003x_1x_2$	2.95	2.91
5	$y = -62.16 + \frac{60519}{x_1} + 0.84x_2$	3.12	3.01
6	$y = 5359.5 + 0.018x_1 + 1404.5x_2 + 0x_1^2 - 3.054x_2^2 - 0.0003x_1x_2 - 1.383\sqrt{x_1} - 16358\sqrt{x_2}$	2.83	2.96

5. З табл. 3 видно, що досліджувані моделі мають від 3 до 8 коефіцієнтів. Найкраща точність спостерігається для квадратичної моделі без взаємодії змінних, що має 5 коефіцієнтів, а найгірша – для лінійної моделі з трьома коефіцієнтами. Порівнюючи теоретичні результати з експериментальними даними (рис. 5) бачимо, що для лінійної моделі характерне заниження прогнозованого показника для автомобілів з дуже високою та з дуже низькою паливною ефективністю. Для квадратичної моделі такі ефекти не спостерігаються – нев'язка майже рівномірно розподілена поміж автомобілів з різною паливною ефективністю.

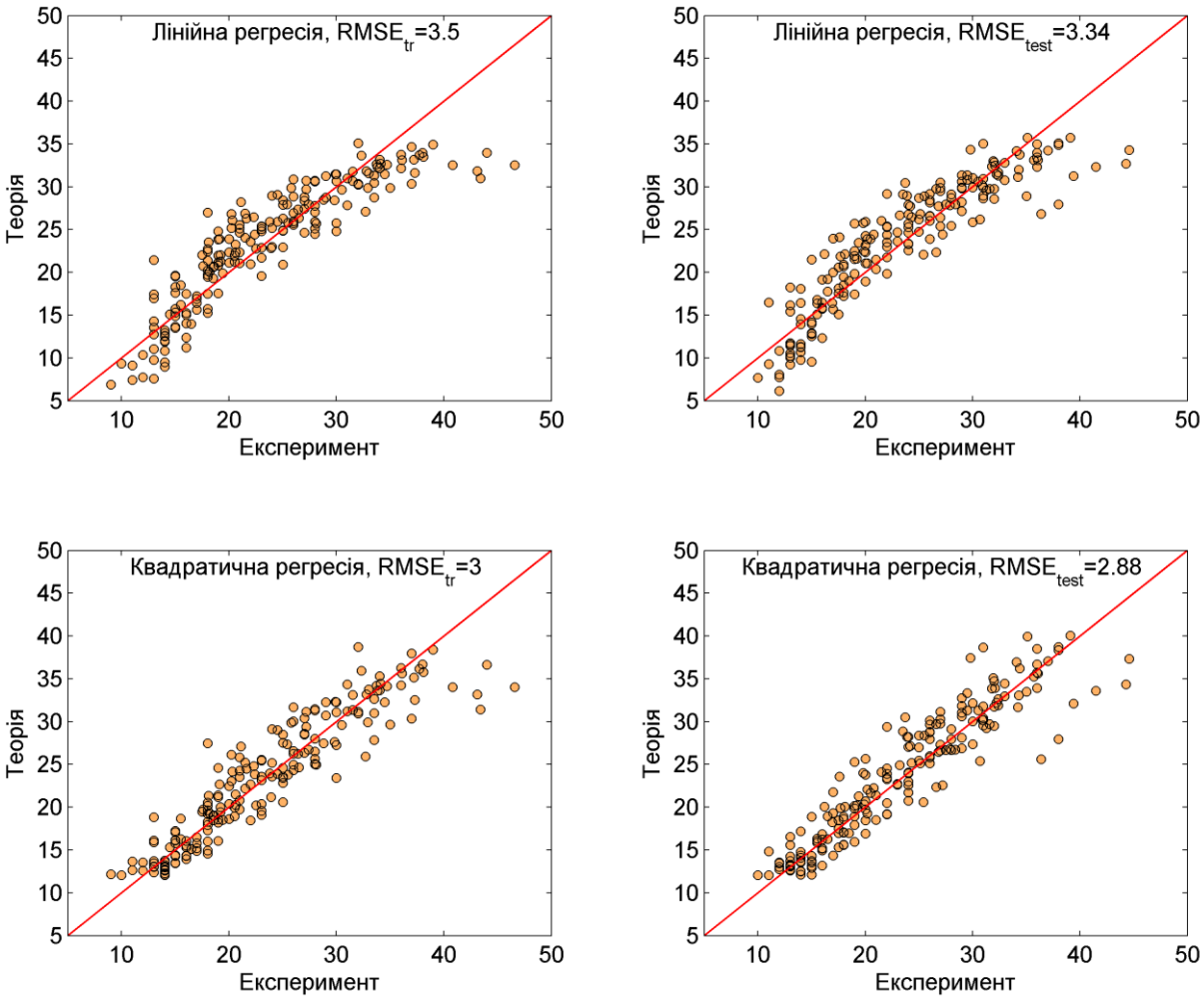


Рисунок 5 – Порівняння експериментальних значень паливної ефективності з результатами моделювання

Поверхня обраної регресійної моделі а також відхилення теоретичних результатів від експериментальних даних наведено на рис. 6. З нього видно, що модель не суперечить здоровому глузду – вона не містить горбів чи провалів, які неможливо змістовно інтерпретувати. Для побудови рис. 6 застосовано функції **`surf`**, **`plot3`** та **`scatter3`**.

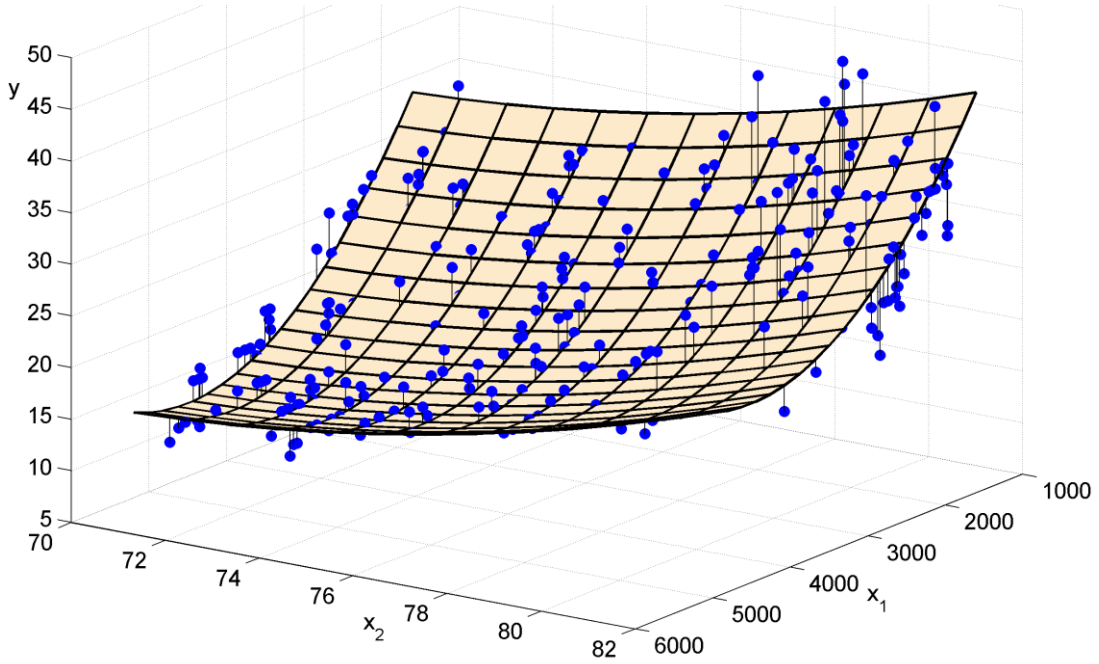


Рисунок 6 – Поверхня “входи – вихід” найкращої регресійної моделі та її відповідність експериментальним даним

6. В табл. 4 порівнюється розроблена модель з трьома конкурентними моделями. Ця таблиця засвідчує, що розроблена модель за точністю незначно поступається лише одній моделі на основі нечітких правил Мамдані, але за складністю значно її переважає.

Таблиця 4 – Порівняння з конкурентами

Модель	Кількість входів	Кількість коефіцієнтів	$RMSE_{tr}$	$RMSE_{test}$	Інтерпрета-бельність
Власна квадратична	2	5	3	2.88	Висока
Лінійна [22]	6	7	3.45	3.44	Висока
Нечітка модель Сугено [22]	2	20	2.67	2.98	Низька
Нечітка модель Мамдані [22]	2	11	2.8	2.88	Висока

Рекомендації

1. Для поглибленого вивчення матеріалу рекомендуємо літературу [6, 7, 11, 28].

2. Регіональні налаштування операційної системи, на якій була створена вибірка даних можуть відрізнятися від налаштувань операційної системи, на якій проводяться лабораторні заняття. Більшість проблем імпорту даних вирішується якщо розділовий знак в десяткових дробах “,” замінити знаком “.”.

3. Під час виконання лабораторної роботи доцільно використовувати такі функції:

sum – сума координат вектора; для матриці – сума елементів по кожному стовпчику;

mean – середнє арифметичне координат вектора; для матриці – середнє арифметичне по кожному стовпчику;

std – середньоквадратичне відхилення координат вектора; для матриці – середньоквадратичне відхилення елементів по кожному стовпчику;

sqrt – корінь квадратний;

norm – норма вектору (за замовчуванням Евклідова відстань);

max – значення та порядковий номер максимального елементу масива;

min – значення та порядковий номер мінімального елементу масива;

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

unique – вилучення з масива дублюючих елементів;

setdiff – різниця множин;

plot – побудова двовимірних графіків за точками;

subplot – розбиття графічного вікна на комірки;

legend – підписування графіків;

xlim – встановлення межі відображення графіки по вісі абсцис;

ylim – встановлення межі відображення графіки по вісі

ординат;

`zlim` – встановлення межі відображення графіки по вісі аплікат;

`xlabel` – підписування вісі абсцис;

`ylabel` – підписування вісі ординат;

`zlabel` – підписування вісі аплікат;

`title` – виведення заголовку графічного вікна;

`hold on` – продовження виведення графіки в поточне вікно без затирання попередніх зображень;

`plot3` – побудова трьохвимірних графіків за точками;

`surf` – побудова зафарбованих поверхонь;

`scatter3` – виведення точок в трьохвимірному просторі;

`regress` – розрахунок коефіцієнтів багатовимірної лінійної регресії;

`polyfit` – розрахунок коефіцієнтів одновимірної поліноміальної регресії.

Детальний опис функції `regress` наведено в **Довіднику ключових функцій**.

Питання для самоконтролю

1. Що таке регресійний аналіз?
2. Яке відношення до регресійного аналізу має К. Гаус?
3. Чому регресійна модель називається регресійною?
4. Що таке принцип зовнішнього доповнення?
5. Яким чином можна реалізувати принцип зовнішнього доповнення?
6. Навіщо дані розбивають на тестову та навчальну вибірку?
7. За якими критеріями перевіряють репрезентативність (однорідність) вибірок даних?
8. За якими критеріями слід обирати регресійну модель?
9. Яка складність алгоритму розрахунку коефіцієнтів регресії?
10. Як визначити достовірність коефіцієнтів регресії?
11. Які програмні системи орієнтовані на пошук наукових публікацій в інтернеті?

2. Дослідження впливу розміру дерева рішень на точність класифікатора

Мета – віднайти оптимальний розмір дерева рішень, який забезпечує найкращу точність класифікації для заданого набору експериментальних даних.

Теоретичні відомості

Розглядається задача класифікації (рис. 7), тобто віднесення об'єкта з ознаками (атрибутами) $X = (x_1, x_2, ..., x_n)$ до одного із класів $\{d_1, d_2, ..., d_m\}$. З математичної точки зору класифікація – це відображення виду $X = (x_1, x_2, ..., x_n) \to y \in \{d_1, d_2, ..., d_m\}$. До класифікації зводяться різноманітні задачі прийняття рішень та розпізнавання образів в інженерному проектуванні, військовій справі, менеджменті, політиці, спорті, в медичній, технічній і економічній діагностиці тощо.

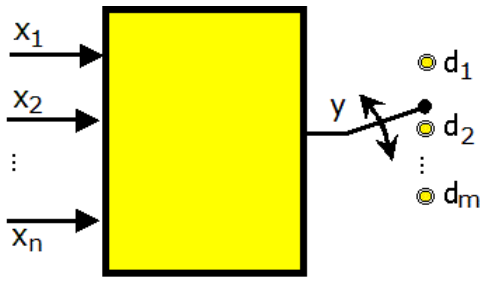


Рисунок 7 – Задача класифікації

Класифікатор будемо створювати лише за вибіркою експериментальних даних:

$$(X_j, y_j), \quad j = \overline{1, M}, \tag{7}$$

де $X_j = (x_{j1}, x_{j2}, ..., x_{jn})$;

$y_j \in \{d_1, d_2, ..., d_m\}$;

M – довжина вибірки.

Якість класифікатора оцінюють за критеріями складності, вартості, інтерпретабельності 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, для задач медичної діагностики – проведення лабораторних досліджень. Тому, намагаються класифікатор розробити таким чином, щоб дорогі атрибути використовувалися рідко. *Інтерпретабельність* 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:

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

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

У вибірці (7) 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 порядкової оцінки використовуються числа.

Класифікатори бувають параметричними та непараметричними. В параметричних класифікаторах наперед визначена модель прийняття рішень, наприклад, деяке рівняння кривої розділу класів. Задача проектування класифікатора полягає в знаходженні таких параметрів цієї моделі, які забезпечують найкращу якість прийняття рішень. В непараметричних класифікаторах структура моделі прийняття рішень визначається не суб'єктивно, а за деяким алгоритмом аналізу вибірки (7). В лабораторній роботі досліджується один із непараметричних класифікаторів – а саме, дерево рішень.

Дерево рішень (decision tree) є однією з найпопулярніших моделей класифікації. Воно являє собою ієрархічний набір правил типу "Якщо – тоді – інакше" у формі орієнтованого дерева. Зазвичай дерево рішень задається бінарним графом, з кожної нетермінальної вершини якого виходять 2 дуги (рис. 8). Корінь дерева є фіктивним; він позначає початок процесу 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. Питання мають такий вигляд "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. Перехід по дереву продовжується до тих пір, поки не потрапимо на один із його листків.

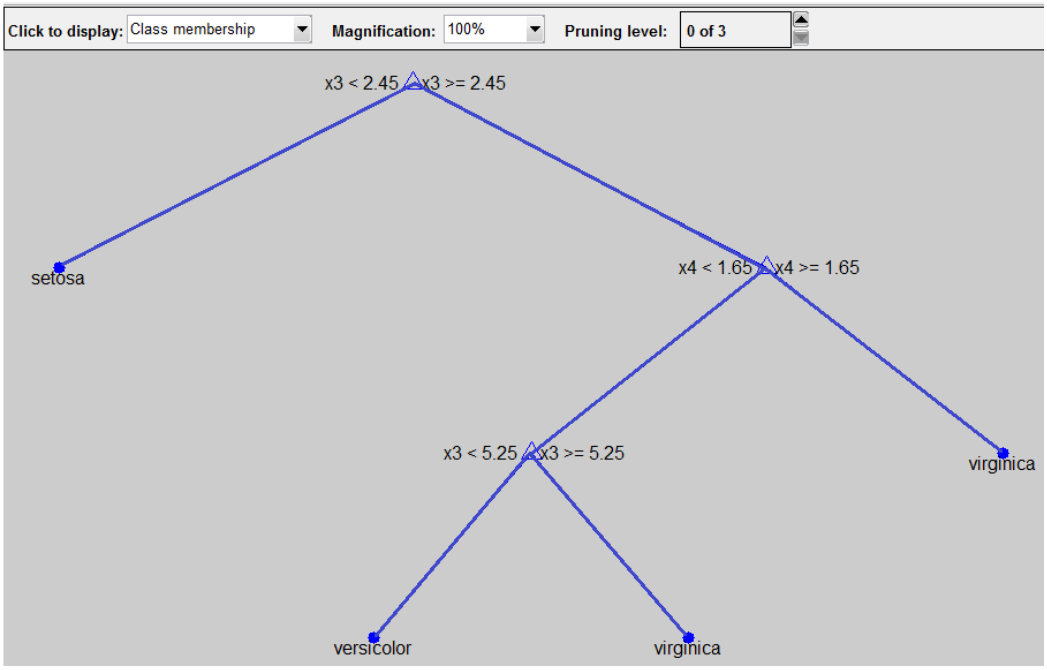


Рисунок 8 – Дерево рішень для класифікації ірисів в форматі MATLAB 7

Для синтезу дерев рішень найчастіше використовуються жадібні алгоритми. Дерево рішень будують від кореня до листків. На кожній ітерації один листок дерева розщеплюється – перетворюється на проміжну вершину з двома новими листками. Логічна умова в новій проміжній вершині формується так, щоб забезпечити екстремальне значення деякого локального критерію ефективності. В логічній умові може фігурувати лише одна із n ознак $x_1, x_2, ..., x_n$. Для кожної із ознак генерується 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 дорівняє 1, що відповідає випадку двох можливих значень змінної x_i. У випадку лише 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 (7) усі значення змінної

x_i мають бути унікальними. 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, що розщеплюється, 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 на принципі мінімізації індексу 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 логічної умови в проміжній вершині дерева. Чим більшим є 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_l та v_r розраховується так:

$$G(v) = \frac{N_l}{N}\left(1 - \sum_{j=\overline{1,m}} p_{lj}^2\right) + \frac{N_r}{N}\left(1 - \sum_{j=\overline{1,m}} p_{rj}^2\right), \qquad (8)$$

де N_l – кількість об'єктів навчальної вибірки, які потрапили до листка v_l;

N_r – кількість об'єктів навчальної вибірки, які потрапили до листка v_r;

N – кількість об'єктів навчальної вибірки, які потрапили до проміжної вершини v, $N = N_l + N_r$;

p_{lj} – частота потрапляння в листок v_l об'єктів з класу d_j, $j = \overline{1, m}$;

p_{rj} – частота потрапляння в листок v_r об'єктів з класу d_j, $j = \overline{1, m}$.

В алгоритмі C4.5 правило розщеплення побудовано на принципі мінімізації ентропії. Ентропія – це міра безпорядку. Для листка дерева рішень абсолютний порядок буде, якщо в нього потрапили об'єкти лише одного класу. Абсолютний безпорядок буде, коли в листку всі класи рішень представлені рівномірно. Ідея алгоритму C4.5 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 з класу d_j у вибірці, $j = \overline{1, m}$;

Далі розрахуємо ентропію кожного варіанту логічної умови:

$$I(v) = -\frac{N_l}{N} \sum_{j=\overline{1, m}} p_{lj} \cdot \log_2(p_{lj}) - \frac{N_r}{N} \sum_{j=\overline{1, m}} p_{rj} \cdot \log_2(p_{lj}).$$

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 d_1 і 100 – класу d_2. Можливі 2 варіанти логічної умови в вершині v, розподіл класів за якими наведено в табл. 5.

Таблица 5 – Розподіл класів по листкам

Варіант логічної умови	v_l		v_r	
	d_1	d_2	d_1	d_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 рівень забруднення 1:10 в вершині v_l, та приблизно 2:1 в вершині v_r. За другим варіантом логічної умови отримуємо рівень забруднення 1:18 в вершині v_l, та 9.5:1 в вершині v_r. За здоровим глуздом зрозуміло, 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 результатів розрахунків видно, що $G(v_{\mathrm{II}}) < G(v_{\mathrm{I}})$, тому, як і очікувалось, другий варіант є кращим.

В алгоритмі C4.5 вибір логічної умови для розщеплення вершини здійснюється за ентропійним підходом. 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:

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

Ентропія для другого варіанта логічної умови дорівнює:

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

Зменшення ентропії для першого варіанта становить:

$\Delta I(v_{\mathrm{I}})=1-0.7906=0.2094.$

Зменшення ентропії для другого варіанта становить:

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

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

Алгоритм синтезу дерева рішень збільшує кількість правил 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 в 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 зазвичай виходить громіздким та переускладненим. 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. 9. На навчальної вибірці залежність є монотонно спадною, тоді як на тестовій вибірці спостерігається екстремум. За принципом зовнішнього доповнення моделлю класифікації слід обрати дерево рішень з мінімальною частотою помилки класифікації на тестовій вибірці. На рис. 9 це дерево позначено літерою “A”.

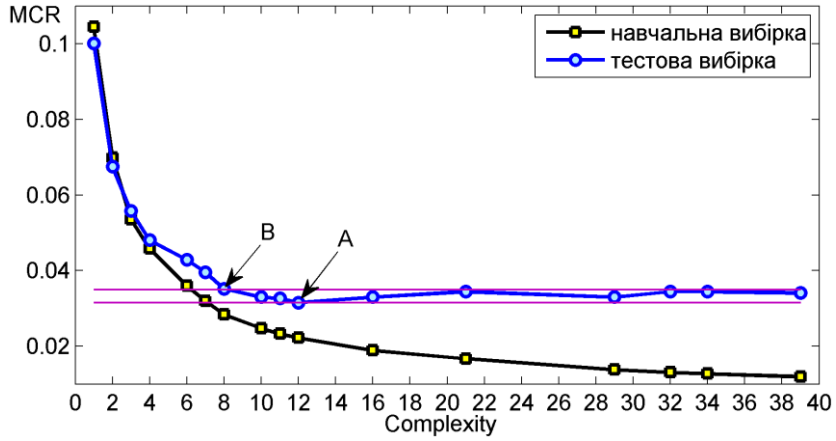


Рисунок 9 – Залежність частоти помилкової класифікації від кількості листків дерева для задачі Page Blocks Classification

За рис. 9 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. 9 є не абсолютно достовірними, бо значення атрибутів містять деякі шуми. Крім того, криві на рис. 9 чутливі до способу розбиття експериментальних даних на навчальну та тестову вибірки. За іншої тестової вибірки частота помилок класифікації могла бути іншою. Відповідно на рис. 9 частоти помилкової класифікації оцінено з деякою похибкою. Ця похибка розраховується таким чином:

$$\Delta = \sqrt{\frac{MCR \cdot (1 - MCR)}{M_{test}}}, \qquad (9)$$

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

За формулою (9) на рис. 9 проведено Δ-коридор, в межах якого класифікатори є статистично нерозрізненими за критерієм точності. Звичайно, серед моделей однакової точності слід обрати найпростішу, якою є класифікатор "B" на рис. 9. Початкове та оптимальне дерева рішень наведені на рис. 10 та 11. Аналізуючи ці класифікатори бачимо, що оптимальне дерево не тільки забезпечує кращу безпомилковість, але і має в 5 раз менше логічних умов. Крім того, для класифікації за оптимальним деревом рішень потрібно значення лише 4 ознак, тоді як за початковим деревом – 9. Таким чином, оптимальне дерево краще за критеріями точності, складності, вартості та інтерпретабельності.

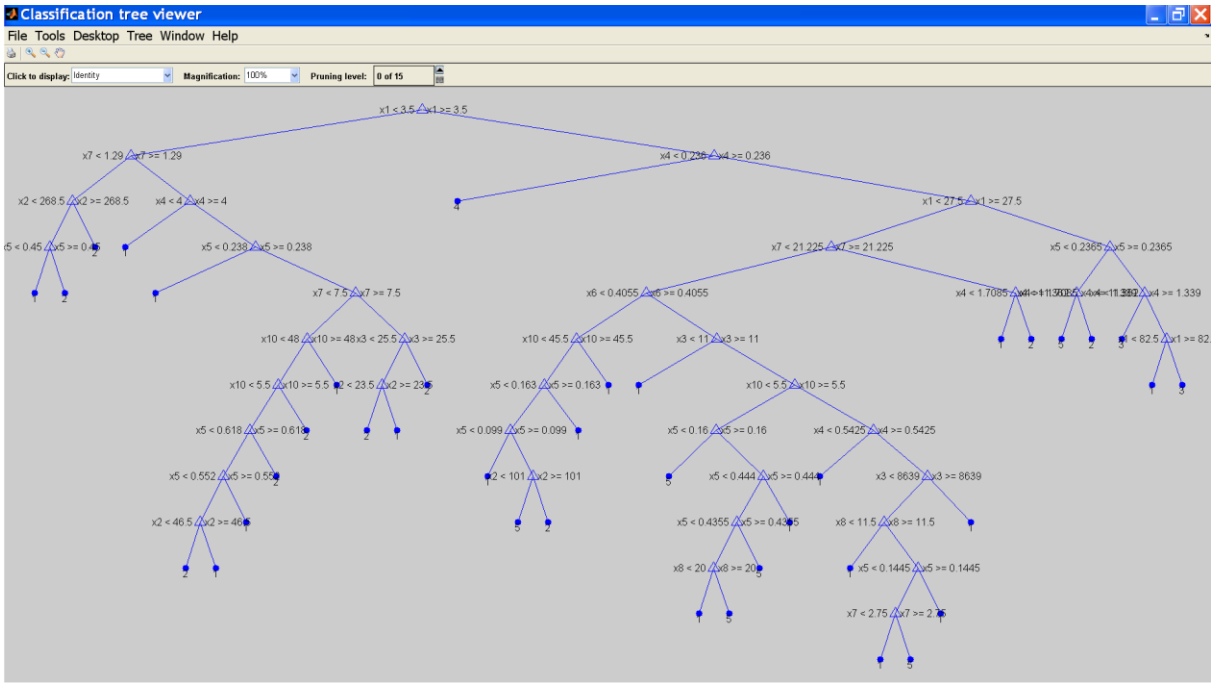


Рисунок 10 – Початкове дерево рішень для задачі Page Blocks Classification в головному вікні модуля Classification tree viewer

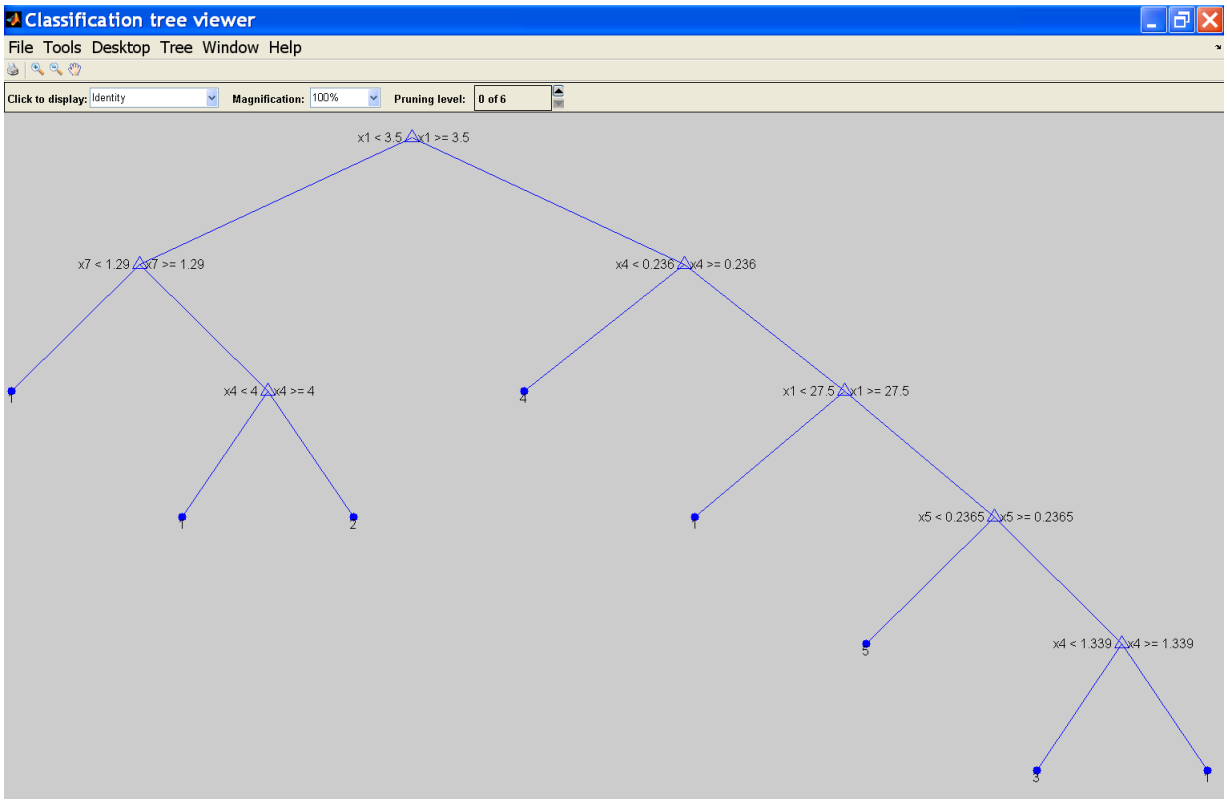


Рисунок 11 – Оптимальне дерево рішень для задачі Page Blocks Classification в головному вікні модуля Classification tree viewer

Для вибору кращого дерева рішень критерій *MCR* доцільно застосувати 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).

Для врахування помилок різних типів під час навчання класифікаторів використовуються 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.

Платіжною називається наступна квадратна матриця:

$$\mathbf{C} = \begin{bmatrix} 0 & c(d_1, d_2) & \dots & c(d_1, d_m) \\ c(d_2, d_1) & 0 & \dots & c(d_2, d_m) \\ \vdots & & & \\ c(d_m, d_1) & c(d_m, d_2) & \dots & 0 \end{bmatrix}, \qquad (10)$$

де $c(d_i, d_j)$ – ціна помилки типу $d_i \to d_j$, коли замість вірного рішення d_i під час класифікації помилково обирається рішення d_j, $i = \overline{1, m}$, $j = \overline{1, m}$, $i \neq j$. Нулі на головній діагоналі матриці (10) вказують на відсутність плати за правильну класифікацію.

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

$$\mathbf{P} = \begin{bmatrix} p(d_1, d_1) & p(d_1, d_2) & \dots & p(d_1, d_m) \\ p(d_2, d_1) & p(d_2, d_2) & \dots & p(d_2, d_m) \\ \vdots & & & \\ p(d_m, d_1) & p(d_m, d_2) & \dots & p(d_m, d_m) \end{bmatrix}, \qquad (11)$$

де $p(d_i, d_j)$ – ймовірність події $d_i \to d_j$, $i = \overline{1, m}$, $j = \overline{1, m}$.

За відомих матриць (10) та (11) ризик розраховується таким чином:

$$R = \sum_{i=\overline{1, m}} \sum_{j=\overline{1, m}} p(d_i, d_j) \cdot c(d_i, d_j).$$

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 безпомилковості класифікатора. Крім того, можливий “зсув” моделі, коли об’єкти з пропусками знаходяться в іншій зоні факторного простору у порівнянні з об’єктами без пропусків. Наприклад, в деяких містах опитування здійснювалося за старими анкетами, а в деяких - за новими, з додатковим питанням. 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 в одному рядку вибірки даних є пропуски. Тоді виходить, що треба вилучити усі ознаки. Можлива і інша ситуація. Зібрана велика вибірка, наприклад, результати анкетування 10 000 осіб. І в одній анкеті респондент не відповів на деяке питання, наприклад, не вказав свій вік. І що, через 1 пропуск проігнорувати відповіді 9 999 респондентів? – очевидно, що шкода втрачати такі дані.

Третій варіант – замінити пропуски на значення "невідомо", тобто розширити шкалу вимірювань. Такий варіант можна застосувати, якщо пропуск стосується ознаки, яка оцінюється за категоріальною шкалою.

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

Популярність застосування дерев рішень для вирішення практичних задач обумовлена їх наочністю і зрозумілістю, простотою процедури класифікації та можливістю використання як числових, так і порядкових та категоріальних атрибутів. Перевагою також є те, що синтез та оптимізація дерев рішень відбувається достатньо швидко навіть для великих вибірок даних. При цьому, під час ідентифікації 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, границі розділу класів яких є непрямокутними, застосування дерев рішень є недоцільним. Одним із шляхів підвищення точності класифікації є застосування не одного класифікатора, а ансамблю (колективу) класифікаторів, за яким можна реалізувати довільні границі розділу класів. Існує 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 компетентності. 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 класу, за який проголосувало найбільше класифікаторів.

Одним із найбільш ефективним методів синтезу ансамблю класифікаторів є бустинг дерев рішень. Таким чином, в ансамблі усі класифікатори є деревами 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. Дерева 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 алгоритму. Підвибірка формується випадково, але так, щоб більше шансів потрапити в неї мали проблемні об’єкти навчальної вибірки. 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 зменшуються 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 точність класифікатора – це не єдиний критерій вибору моделі прийнятті рішень. При збільшенні ансамблю погіршуються інші критерії якості моделі – збільшується 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, що призводить, наприклад, до необхідності 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 репозиторію автоматичного навчання Каліфорнійського університету в Ірвіні [23]. Лабораторна робота полягає у виконанні таких завдань.

1. Сформулювати змістовну постановку задачі згідно варіанту з табл. 6.
2. Розбити дані на навчальну та тестову вибірки та перевірити їх репрезентативність.
3. Синтезувати дерево рішень для здійснення класифікації об’єктів для задачі згідно до табл. 6.
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 двохвимірні 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 6 – Варіанти завдання

Варіант	Задача
1	Adult
2	Bank Marketing
3	Car Evaluation Data Set
4	Cardiotocography

Варіант	Задача
5	Chess (King-Rook vs. King)
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
17	Spambase
18	Stalog (Shuttle)
19	Statlog (German Credit Data)
20	Statlog (Vehicle Silhouettes)
21	Steel Plates Faults
23	User Knowledge Modeling Data Set
24	Yeast

Рекомендації

Для поглибленого вивчення матеріалу рекомендуємо літературу [1–4, 12, 25, 28].

Під час виконання *першого завдання* бажано не лише перекласти опис задачі, який наведено в [23], але навести більше змістовної інформації. Приклад виконання цього завдання для задачі розпізнавання ірисів наведено нижче.

Розглядається задача класифікації квіток ірисів за такими 4-ма ознаками:

x_1 – довжина чашолистка;

x_2 – ширина чашолистка;

x_3 – довжина пелюстки;

x_4 – ширина пелюстки.

Розмір вибірки для цієї задачі складає 150. 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 можливих 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. 12.

Iris-setosa Iris-versicolor Iris-virginica

Рисунок 12 – Фото ірисів з сайту http://www.signa.org

Примітка. Регіональні налаштування операційної системи, на якій була 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 відкриті файлу MS Excel виникає помилка, спробуйте запустити відповідний програмний додаток з пакету та "перетягнути" файл в середину або відкрити через відповідне меню.

Під час виконання *другого завдання* слід пам'ятати, що для категоріальних змінних перевірку на репрезентативність треба проводити 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 розбиття даних на навчальну та тестові вибірки доцільно

використовувати функції **min**, **max**, **unique** 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 використовувати функцій **import::cvs**, **import::readdata** та **fscanf**.

Для виконання *третього завдання* доцільно використовувати функцію **classregtree** таким чином:

T = classregtree(tr_set_x, tr_set_y),

де **T** – дерево рішень;

tr_set_x – вхідні значення навчальної вибірки;

tr_set_y – вихідні значення навчальної вибірки.

Якщо в даних є пропускі, то у відповідні чарунки матриці **tr_set_x** слід занести **NaN**.

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

T = classregtree(tr_set_x, tr_set_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_set_x, tr_set_y, 'categorical', [1 3]).

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

T = classregtree(tr_set_x, tr_set_y, 'cost', Cost_matrix).

Для виконання *четвертого завдання* слід використовувати функції **prune** – для підрізання дерева рішень (за вершина або за рівнями) та **eval** – для класифікації. В багатьох випадках щоб розрахувати частоти 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. 10 та 11).

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

```
T = classregtree(tr_set_x, tr_set_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:

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

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 якійсь 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. 13.

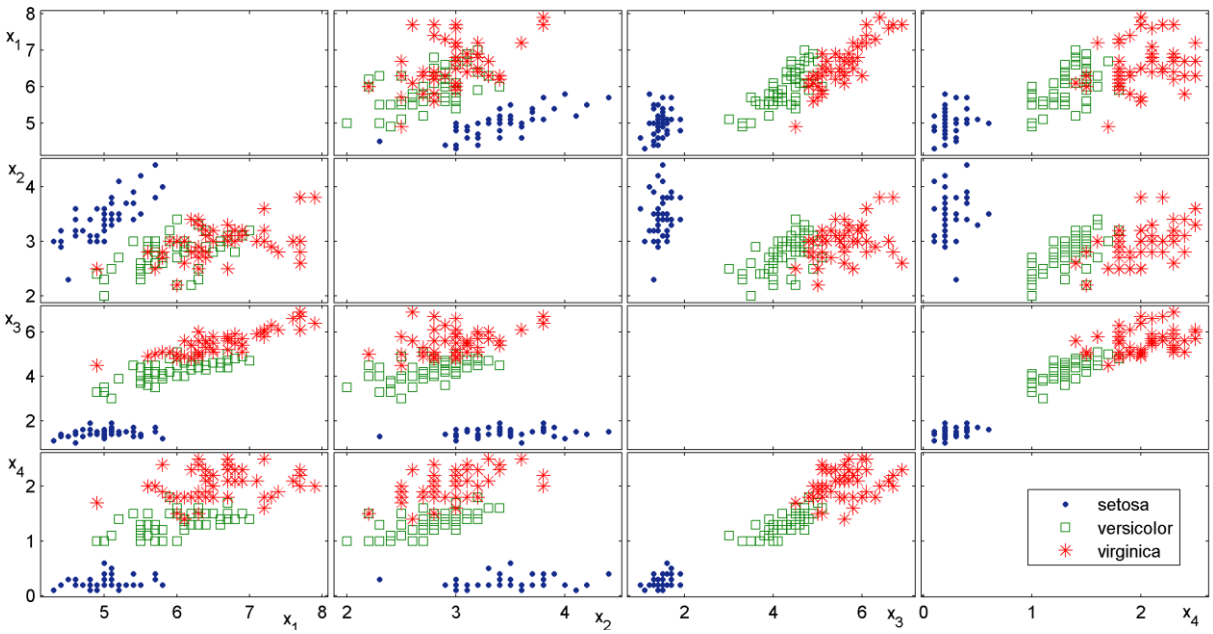


Рисунок 13 – Двофакторні розподіли класів ірисів

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 масива;

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

`setdiff` – різниця множин;

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

unique – вилучення з масива дублюючих елементів;

gplotmatrix – створення графічне 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 Кондерсе 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 існує різних типів помилок для задачі класифікації з трьома класами?
12. Які труднощі визначення платіжної матриці для реальних задач прийняття рішень?
13. Як оцінити середні витрати для здійснення класифікації за деревом рішень, якщо відома вартість вимірювання кожної ознаки?

3. 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, формуванні 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 підстроюють модель, змінюючи її параметри таким чином, щоб найточніше описати залежність в експериментальних даних. Цей етап найбільш формалізований і, зазвичай, зводиться до вирішення задачі оптимізації із мінімізації середньоквадратичної нев’язки для об’єктів з неперервним виходом, чи рівня безпомилковості класифікації для об’єктів з дискретним виходом. У випадку моделювання залежностей за допомогою

нечітких баз знань на етапі параметричної ідентифікації настроюють 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. Відповідно, елементи належать множині з різним ступенем. За такого підходу висловлювання про те, що елемент належить множині втрачає сенс, тому що необхідно вказати наскільки сильно, 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 вище ступінь належності, тим більшою мірою 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: $\tilde{A} = \left(\mu_A(u_1)/u_1,\ \mu_A(u_2)/u_2, ..., \mu_A(u_k)/u_k\right)$. Наприклад, нечітку множину *"чоловік середнього зросту"* можна задати такою нечіткою множиною $\tilde{A} = \left(0.1/160, 0.3/165, 0.8/170, 1/175, 1/180, 0.5/185, 0.1/190\right)$. У випадку неперервної множині U використовують позначення $\tilde{A} = \int_u \mu_A(u)/u$, 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 $\mathrm{supp}(\tilde{A})$.

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

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 $\tilde{C} = \tilde{A} \cap \tilde{B}$ з функцією належності $\mu_C(u) = \mu_A(u) \wedge \mu_B(u)$ для усіх $u \in U$, де $\wedge$ - t-норма (табл. 7).

Об'єднанням нечітких множин $\tilde{A}$ та $\tilde{B}$, які задані на U, називається нечітка множина $\tilde{D} = \tilde{A} \cup \tilde{B}$ з функцією належності $\mu_D(u) = \mu_A(u) \vee \mu_B(u)$ для усіх $u \in U$, де $\vee$ - s-норма (див. табл. 7).

Таблиця 7 – 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

Приклади виконання операцій доповнення, перетину та об'єднання над нечіткими множинами з використанням норм Заде наведені на рис. 14.

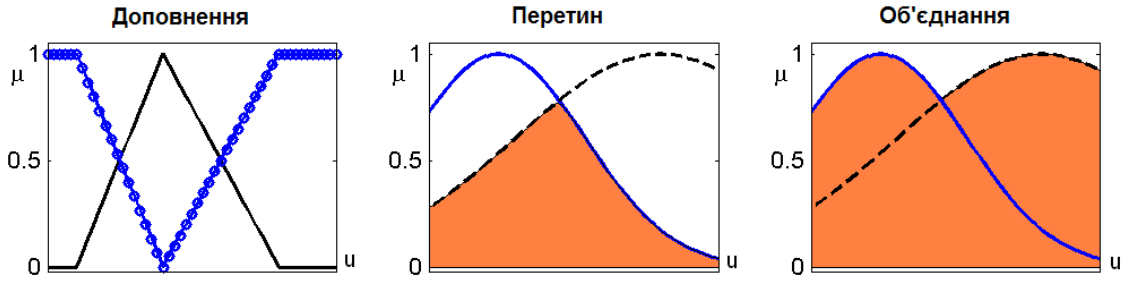


Рисунок 14 – Операції над нечіткими множинами

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 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 оцінками найбільше поширення отримали методи на основі парних порівнянь та статистичного оброблення експертної інформації.

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:

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

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 вектора $W = (w_1, w_2, ..., w_k)^T$ 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 А.

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

$$\tilde{l} = (\mu_l(u_1)/u_1,\ \mu_l(u_2)/u_2, ..., \mu_l(u_k)/u_k),$$

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 аргумента 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. 8.

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 ($\bar{}$) і імплікація ($\Rightarrow$) виконуються за такими правилами:

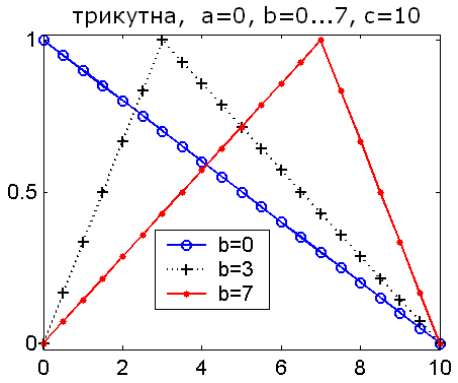
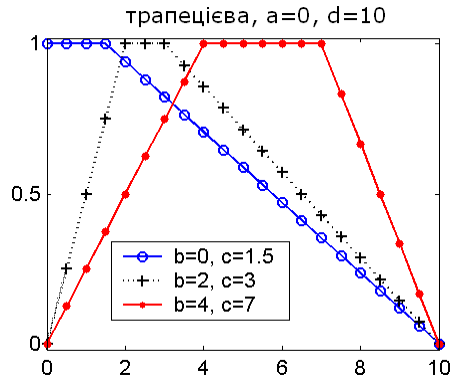
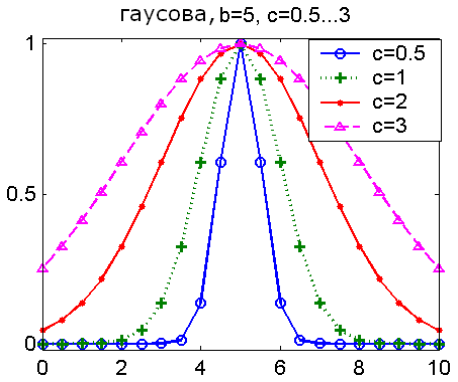
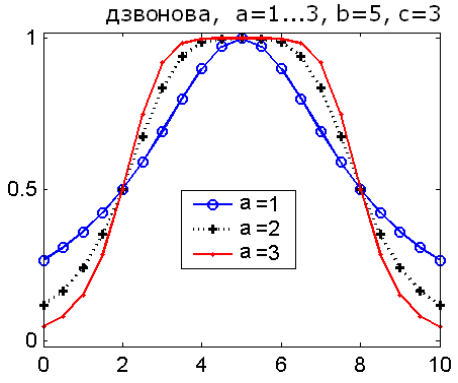
$$\mu_{A \wedge B}(u) = \min\left(\mu_A(u), \mu_B(u)\right);$$

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

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

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

Таблиця 8 – 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.
	$\mu(y)=\dfrac{1}{1+\left\lvert\dfrac{x-b}{c}\right\rvert^{2a}},$ де a – коефіцієнт крутизни; b – координата максимуму; c – коефіцієнт концентрації.

Нечітким виведенням називається апроксимація залежності $y = f(x_1, x_2, \ldots, x_n)$ за допомогою нечітких правил <Якщо – тоді> та нечітких логічних операцій.

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*. Сьогодні 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 моделює залежність $y = f(x_1, x_2, \ldots, x_n)$, запишемо таким чином:

$$\begin{array}{l} \text{Якщо}\left(x_1 = \tilde{a}_{1j} \ \text{ та } \ x_2 = \tilde{a}_{2j} \ \text{ та } \ldots \text{та } \ x_n = \tilde{a}_{nj} \ \text{ з вагою } w_j\right), \\ \text{тоді } \ y = \tilde{d}_j, \quad j = \overline{1, N}, \end{array} \tag{12}$$

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 $\tilde{l}_r = \int\limits_{y \in [\underline{y}, \overline{y}]} \mu_{l_r}(y)/y$, $r = \overline{1, m}$;

$w_j \in [0, 1]$ – 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 (12) зручно подавати у формі табл. 9. Для адекватного опису залежності $y = f(x_1, x_2, \ldots, x_n)$ в деякій області факторного простору не завжди необхідно знати 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 9 – 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 (12) здійснюють за схемою з рис. 15. Спочатку розраховують ступінь виконання антецедента j-го правила для поточного вхідного вектора $X^* = \left(x_1^*, x_2^*, \ldots, x_n^*\right)$:

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

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

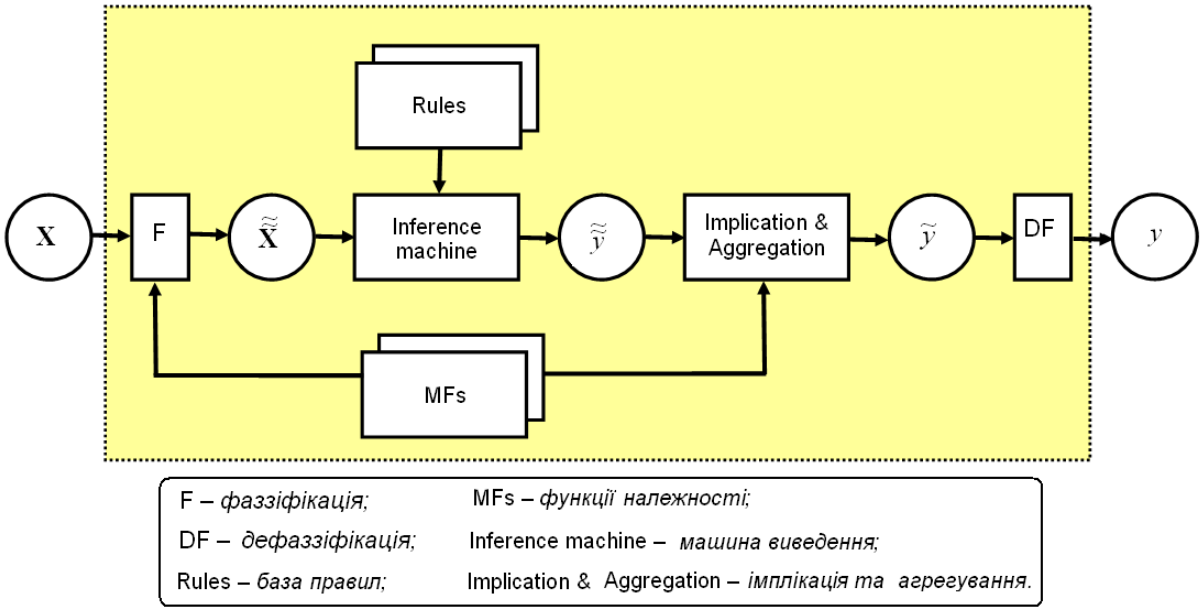


Рисунок 15 – Логічне виведення за нечіткою базою знань Мамдані

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 елементи універсальної множини задані нечіткими множинами $\tilde{d}_1, \tilde{d}_2, ..., \tilde{d}_N$. Для перетворення $\tilde{\tilde{y}}^*$ в звичайну нечітку множину виконаємо такі дії. Спочатку представимо результат виведення за *j*-м правилом бази знань у формі такої нечіткої множина:

$$\tilde{d}_j^* = \text{imp}\left(\tilde{d}_j,\ \mu_j(X^*)\right),\quad j = \overline{1, N}, \tag{13}$$

де imp позначає імплікацію, яку реалізують операцією мінімума.

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 (13):

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

де agg – агрегування нечітких множин, яке реалізують операцією максимума. Ілюстрацією цієї формули є рис. 16, де здійснюється агрегування трьох нечітких множин.

Чітке значення виходу y^*, яке відповідає вхідному вектору X^*, визначається через дефаззіфікацію нечіткої множини $\tilde{y}^*$. Популярні методи дефаззіфікації зведено в табл. 10.

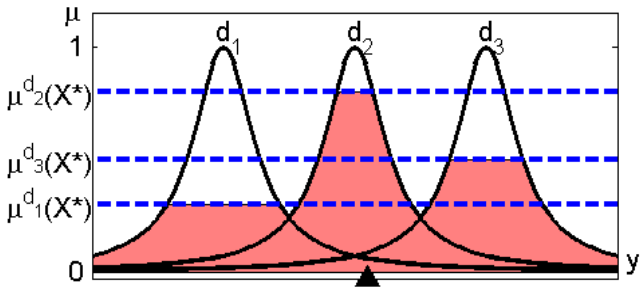


Рисунок 16 – Імплікація, агрегування та дефаззіфікація в алгоритмі Мамдані

Таблиця 10 – Дефаззіфікація різними методами нечіткої множини $\tilde{A} = \int\limits_{u\in[\underline{u},\bar{u}]} \mu_A(u)/u$

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

Лабораторна робота стосується задачі структурної ідентифікації за допомогою нечіткої бази знань. Конкретніше завдання полягає у генеруванні множини нечітких правил, які описують особливості залежності що ідентифікується. З цих правил-кандидатів слід сформувати нечіткі бази знань різної розмірності 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 середньо квадратичної нев'язки ($RMSE$). Для оцінювання складності нечіткої моделі часто використовують такі показники:

N_{rules} – 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, $i=\overline{1,n}$;

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

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

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

Вибір 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 вибору правил бази знань. Аналогічно до класичних постановок задачі про рюкзак сформовано й задачі вибору правил нечіткої бази знань. Задача вибору правил нечіткої бази знань, 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_{rules} нечіткої бази знань Мамдані. Дослідження проведемо для 2 еталонних залежностей (рис. (15)17) – неспадної та унімодальної:

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

$$y = -{x_1}^2 - {x_2}^2,\ x_1 \in [-7;3],\ x_2 \in [-5;5]. \tag{15}$$

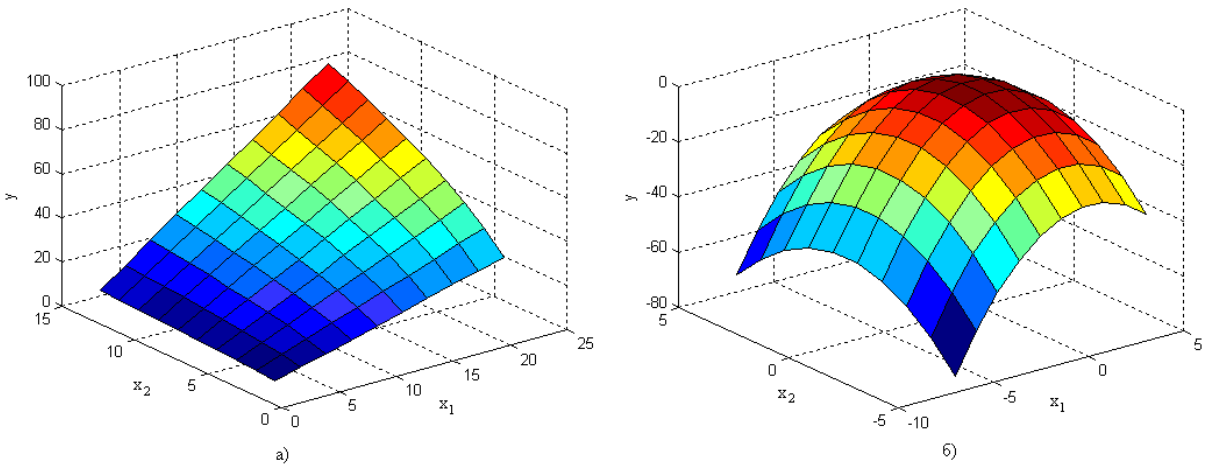


Рисунок 17 – Еталонні залежності:
а) неспадна (14); б) унімодальна (15).

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 проводилися 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 неспадної залежності (14) для оцінювання вхідних змінних використовувалось 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. Для унімодальної залежності (15) використовувалися такі нечіткі розбиття вхідних змінних: 3x3, 3x4, 3x5, 4x3, 4x4 та 5x3. Вихідна змінна в обох задачах оцінена п'ятьма термами.

Результати експериментів (рис. 18 та 19) показали, що усереднена нев'язка $RMSE_{mean}$ спадає зі збільшенням кількості нечітких правил і досягає мінімума за повної бази знань. Графік нев'язки для найкращих комбінацій правил ($RMSE_{min}$) відповідає парето-фронту. Кожна база знань

з парето-фронта є найкращою моделлю з заданим рівнем складності N_{rules}. Для парето-фронта на кривих навчання добре простежується плато насичення, коли додавання нових правил майже не змінює адекватність нечіткої моделі.

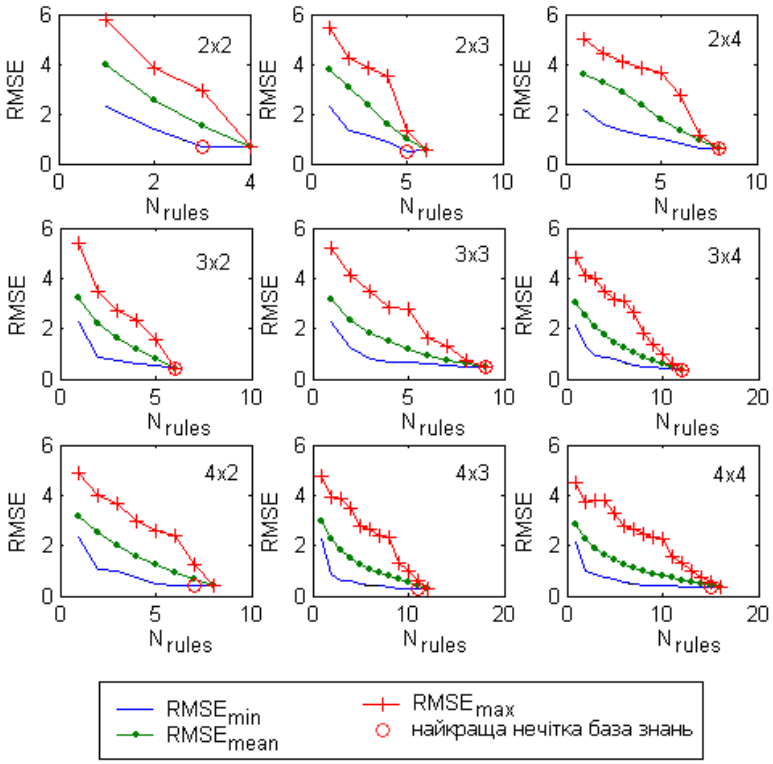


Рисунок 18 – Криві навчання баз знань Мамдані для залежності (14)

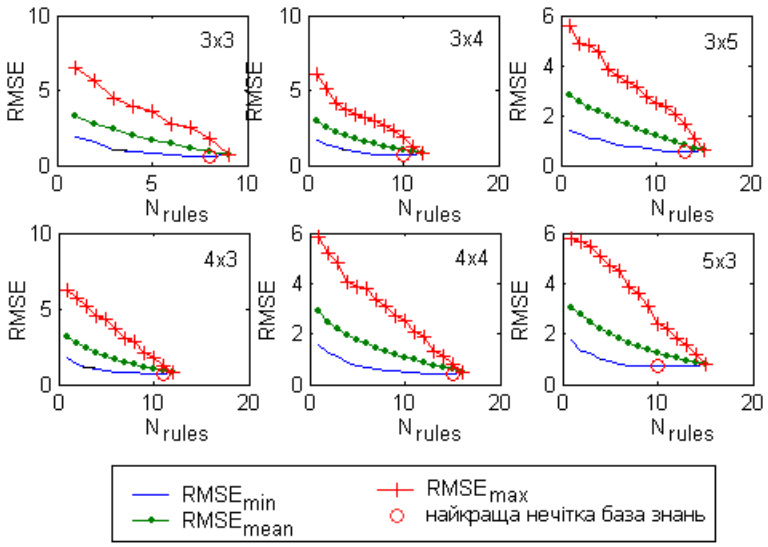


Рисунок 19 – Криві навчання баз знань Мамдані для залежності (15)

Завдання на лабораторну роботу

В лабораторній роботі проектується нечітка база знань Мамдані, яка моделює еталону залежність $y = f(x_1, x_2)$ з табл. 11. 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. 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. Провести узагальнення результатів експериментів, провівши криві навчання $RMSE_{min}$, $RMSE_{max}$ та $RMSE_{mean}$.

5. Побудувати криву навчання у формі залежності $RMSE$ від N, використовуючи жадібний алгоритм вибору 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 формування 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.

Таблиця 11 – 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]	[2, 14]
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, 100]
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, 7]
8	$y = \frac{{x_1}^2 - {x_2}^3}{2x_1 + x_2}$	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]	[2, 7]
12	$y = 7\sqrt{x_1} - 6x_1x_2$	4	3	5	[30, 50]	[50, 100]
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	$y = \frac{x_1 + {x_2}^3}{2\sqrt{x_1x_2}}$	3	4	5	[2, 7]	[10, 20]
16	$y = 0.01{x_1}^2 tg(x_2)$	5	4	4	[-20, 25]	[-1, 1.3]
17	$y = (x_1 + x_2)\sin(x_1 + x_2)$	4	3	5	[1, 4]	[0,5, 2.5]
18	$y = \frac{{x_1}^2 + x_2}{e^{x_1x_2}}$	4	4	5	[-4, -3]	[0, 1]
19	$y = \sin(x_1)\cos(2x_2)$	4	5	4	[1, 2]	[1, 2]

Варі-ант	Аналітична залежність	Кількість термів			Діапазон	
		x_1	x_2	y	x_1	x_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	$y=(x_1x_2)\sqrt{x_1^2+x_2^2}$	4	4	5	[5, 20]	[-10, 10]
24	$y=(x_1-x_2)(x_1^2+x_1x_2+x_2^2)$	4	5	4	[0.1, 2]	[0, 1.5]

Рекомендації

Для поглибленого вивчення матеріалу рекомендуємо літературу [13–15, 18–22]. Доцільно ознайомитись з описом GUI-модулів пакету Fuzzy Logic Toolbox з [22].

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 показані на рис. 20.

```
%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); %цільова функція
%Створюємо декартову сітку nxn, в якій обрахуємо значення:
[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')
```

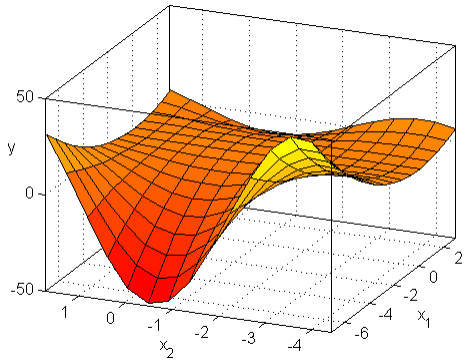


Рисунок 20 – Графік функції $y = x_1^2 \cdot \sin(x_2 - 1)$

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. 21.

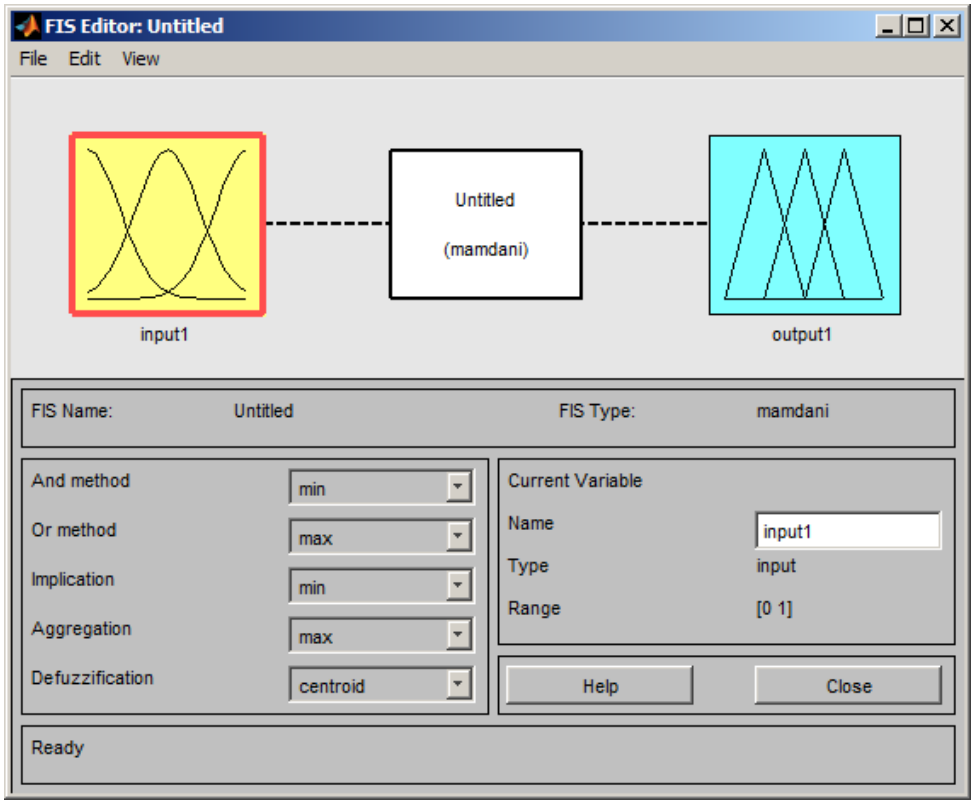


Рисунок 21 – Головне вікно fis-редактору

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

Крок 3. Перейменувати першу вхідну змінну. Для цього слід зробити одне натиснення лівої кнопки 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 зробити одне натиснення лівої кнопки 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 зробити одне натиснення лівої кнопки 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 робимо одне натиснення лівою кнопкою миші на графіку першої функції належності. (див. рис. 22). Потім вводимо найменування терму, наприклад, **Низький**, в полі Name і натискаємо <Enter>. Потім робимо одне натиснення лівою кнопкою миші на графіку другої функції належності і вводимо найменування терму, наприклад, **Середній**, в полі Name і натискаємо <Enter>. Ще раз робимо одне натиснення лівою кнопкою миші по графіку третьої функції належності і введемо найменування терму, наприклад, **Високий**, в полі Name і натискаємо <Enter>. В результаті отримуємо графічне вікно, яке зображено на рис. 22.

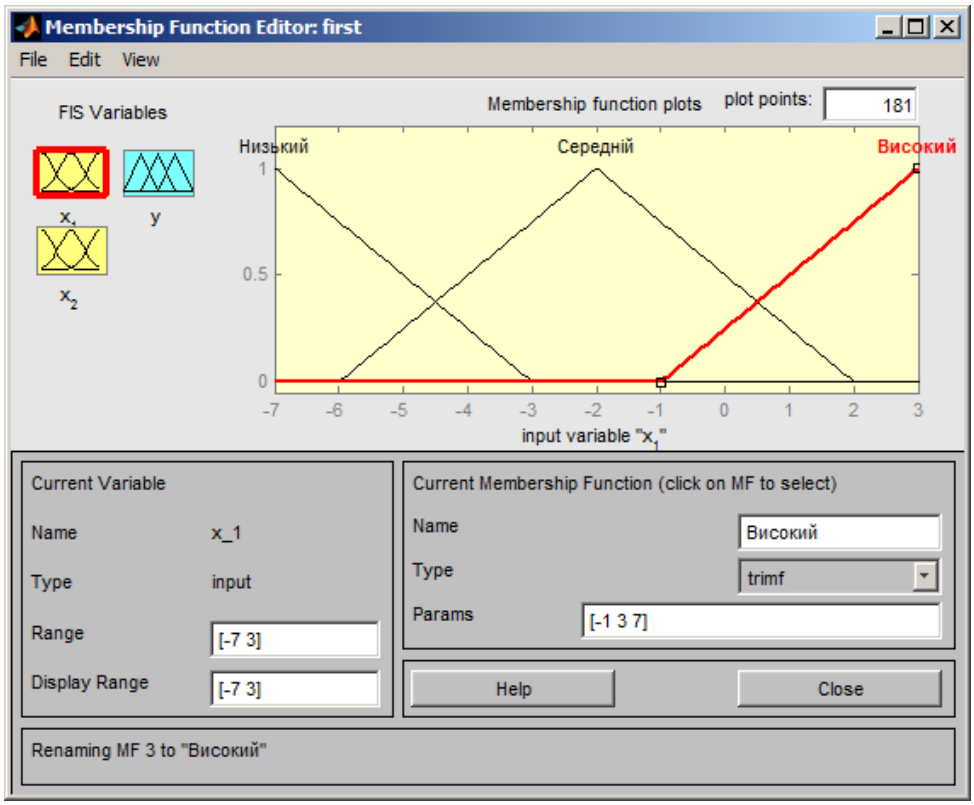


Рисунок 22 – Функції належності змінної x_1

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 терм Середній займав всю область визначення, а два крайніх терми перетинались посередині. В результаті отримуємо графічне вікно, яке зображене на рис. 23.

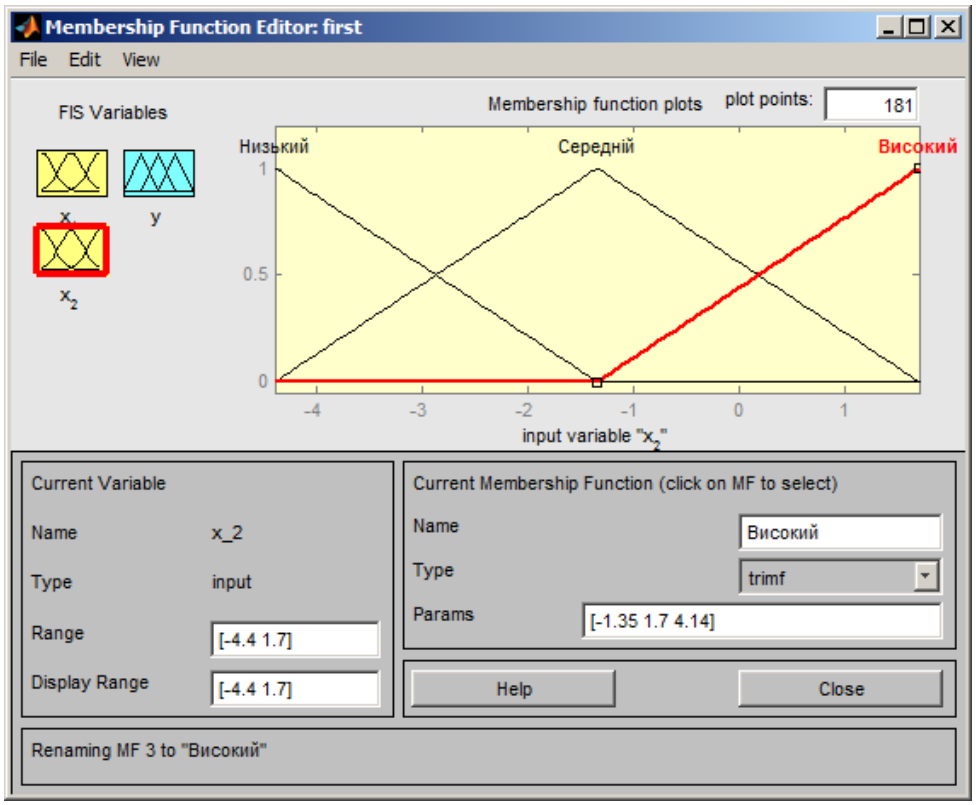


Рисунок 23 – Функції належності змінної x_2

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 (див. рис. 24) і натиснемо <Enter>. В меню Edit оберемо команду Add MFs.... Потім 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. 24.

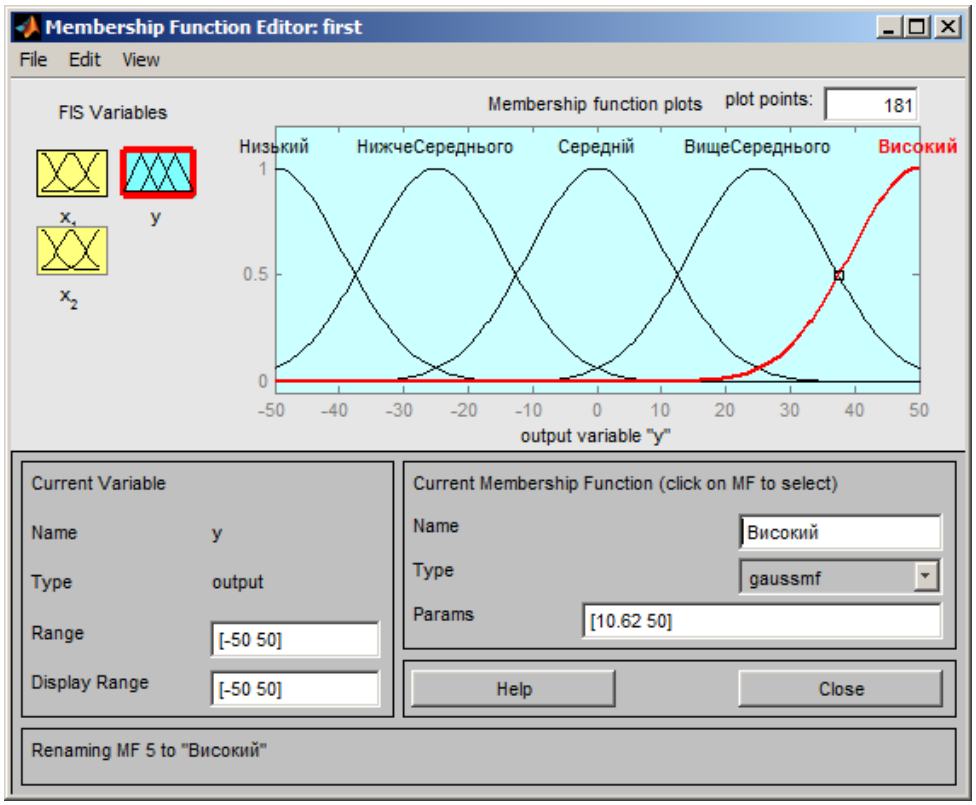


Рисунок 24 – Функції належності змінної y

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. 20, 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 необхідно обрати в меню відповідну комбінацію термів і натиснути кнопку Add rule. На рис. 25 зображено вікно редактору бази знань після введення усіх 7 правил. В кінці кожного правила в дужках наведено ваговий коефіцієнт. Ваговий коефіцієнт можна задати в полі в Weight.

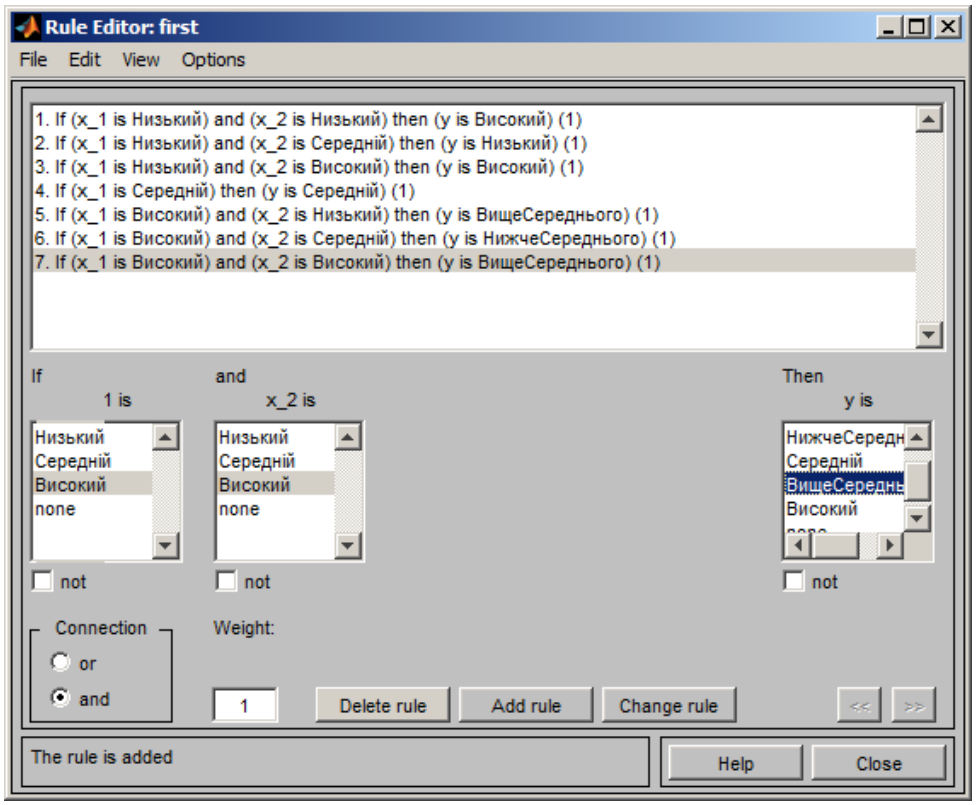


Рисунок 25 – Вікно правил

Крок 17. Збережемо створену нечітку систему. Для цього в меню File в підменю Export оберемо команду To File….

На рис. 26 наведено вікно візуалізації нечіткого логічного виведення. Воно активується командою Rules меню View. В полі Input вказуються значення вхідних змінних, для яких виконується логічне виведення. Тобто, обраховується за алгоритмом Мамдані значення вихідної змінної.

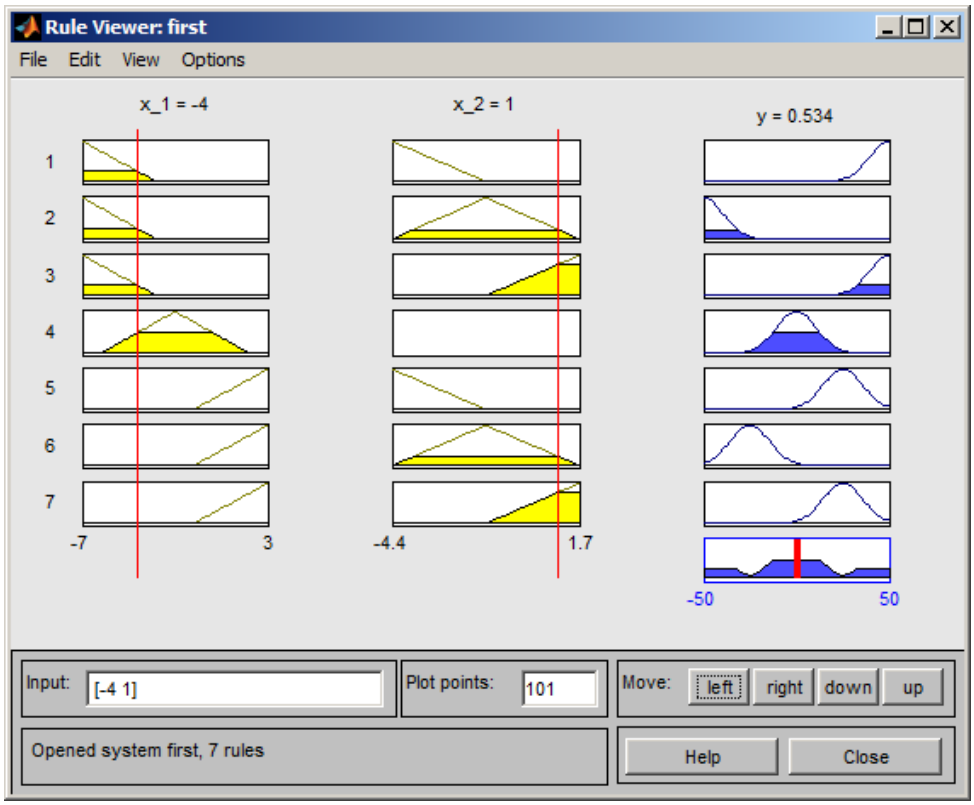


Рисунок 26 – Візуалізація нечіткого логічного виведення

На рис. 27 зображена поверхня “входи – вихід”, яка відповідає синтезованій нечіткій системі. Для виводу цього вікна необхідно в меню View обрати команду Surface. Порівнюючи поверхні на рис 20 і рис. 27, робимо висновок, що 7 нечітких правил досить добре описують складну нелінійну залежність.

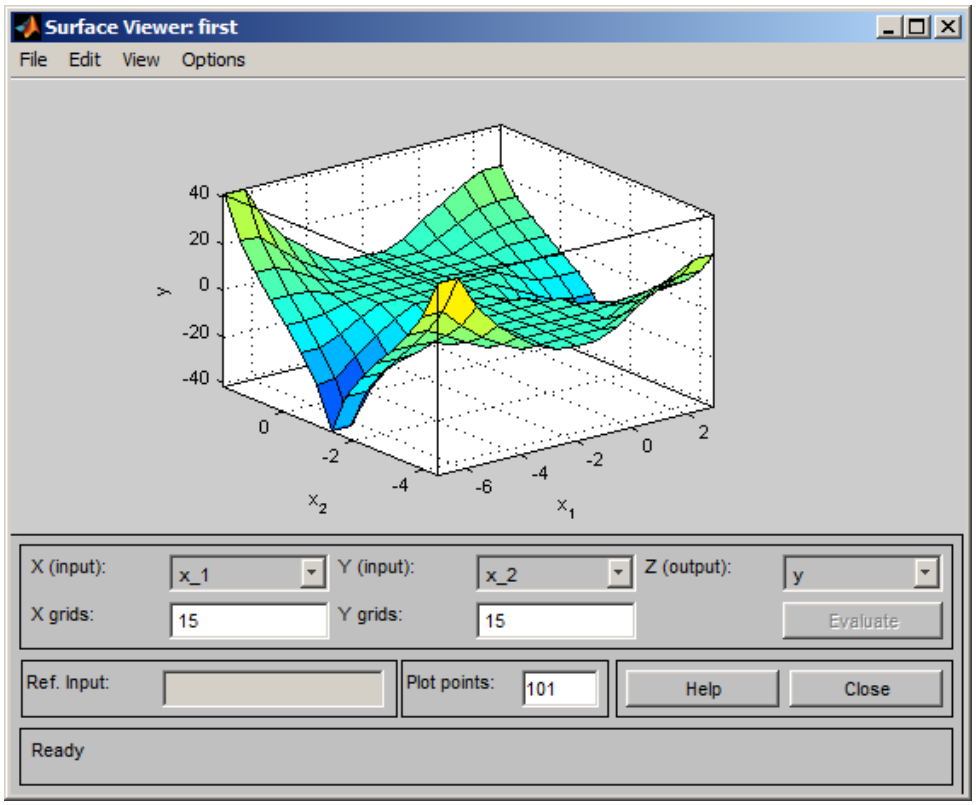


Рисунок 27 – Поверхня “входи – вихід” нечіткої системи

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

Для виконання *четвертого*, *п’ятого* чи *шостого завдань* недоцільно застосовувати громіздкі конструкції з 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 вплине на логічне виведення. Це еквівалентно вилученню правила з бази знань. Для активації правила слід присвоїти ваговому коефіцієнту одиничне значення. Поле вагового коефіцієнта *i*-го правила нечіткої системи, яка представлена структурою **`fis`**, вказується таким чином: **`fis.rule(i).weight`**. Розгорнута схема fis-структури наведена на рис. 28.

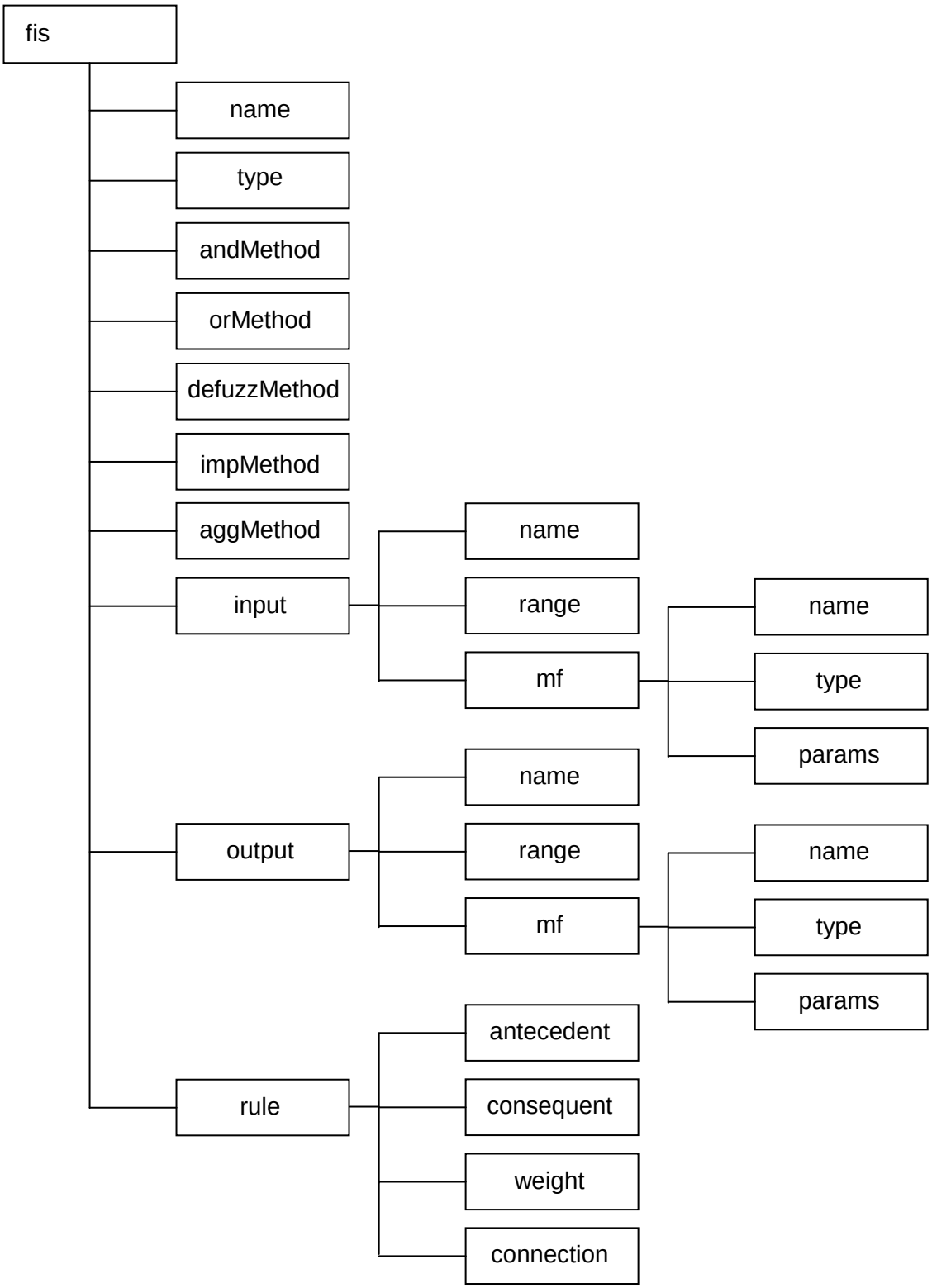


Рисунок 28 – Fis-структура

Поля fis-структури призначені для збереження наступної інформації:

`name` – найменування системи нечіткого логічного виведення;

`type` – тип системи (допустимі значення **`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 дефаззіфікації (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** – ймовірносне 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 – модель функції належності вхідної змінної (запрограмовані моделі: **dsigmf** – функція належності у формі різниці двох сигмоїдних функцій; **gauss2mf** – двохбічна гаусова функція належності; **gaussmf** – гаусова функція належності; **gbellmf** – узагальнена дзвонова функція належності; **pimf** – пі-подібна функція належності; **psigmf** – добуток двох сигмоїдних функцій належності; **sigmf** – сигмоїдна функція належності; **smf** – s-подібна функція належності; **trapmf** – трапецієва функція належності; **trimf** – трикутна функція належності; **zmf** – z-подібна функція належності);

input.mf.params – масив параметрів функції 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 – модель функції належності вихідної змінної (запрограмовані моделі для системи типу Мамдані: **dsigmf** – функція належності у формі різниці двох сигмоїдних функцій; **gauss2mf** – двохбічна гаусова функція належності; **gaussmf** – гаусова функція належності; **gbellmf** – узагальнена дзвонова функція належності; **pimf** – пі-подібна функція належності; **psigmf** – добуток двох сигмоїдних функцій належності; **sigmf** – сигмоїдна функція належності; **smf** – s-подібна функція належності; **trapmf** – трапецієва функція належності; **trimf** – трикутна функція належності; **zmf** – z-подібна функція належності);

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

rule – масив правил нечіткої бази знань;

rule.antecedent – антецедент правила (вказуються номери термів у порядку запису вхідних змінних. Число 0 вказує на те, що значення відповідної вхідний змінної не впливає на істинність правила);

rule.consequent – консеквент правила (вказуються номери термів у порядку запису вихідних змінних. Число 0 вказує на те, що правило не поширюється на відповідну вихідну змінну);

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

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

Виконання *шостого* завдання потребує значних обчислювальних ресурсів. Для бази знань з 16-ти правил перебір усіх варіантів за тестової вибірки з 100 точок займе до 10 хвилин. Якщо додати ще 3 правила, тоді тривалість експерименту може зрости до кількох годин. Якщо додати 5-6 правил експеримент триватиме кілька днів.

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

```
% RMSEfis - функція для обрахунку RMSE, яку потрібно написати
%           самостійно.
% INP - вхідні значення тестової вибірки.
% OUT - вихідні значення тестової вибірки.
fis = readfis('first');      % зчитуємо збережену базу знань.
Nrules = length(fis.rule);
for k=1:Nrules
    N=nchoosek(Nrules,k);    % повертає кількість комбінацій.
```

```
    M=nchoosek(1:Nrules,k);    % повертає матрицю всіх комбінацій
                               % k правил.
    Error{k} = zeros(N,1) );  % виділяємо місце під обрахунки.
    for i=1:N
        fis.rule(:).weight=0; % правила - в початковий стан.
        fis.rule( M(i,:) ).weight=1;
        Error{k}(i)=RMSEfis(fis, INP, OUT);%обраховуємо помилку.
    end
end
```

Примітка. Згідно з документацією, функцію **nchoosek** слід застосовувати для перебору не більше 15 елементів через стрімке зростання тривалості обчислювальних експериментів.

Якщо кількість правил-кандидатів дорівнює $N_{\max}$, тоді можна згенерувати $(2^{N_{\max}}-1)$ унікальних баз знань. Усі ці варіанти комбінацій правил можна отримати за допомогою швидких операцій побітового зсуву **bitshift** та перевірки по модулю 2 **mod**. Програмна реалізація такого генерування нечітких баз знань наведена нижче:

```
% rulecount - кількість правил в базі знань.
for i = 1:2^rulecount-1
    rulenum = 0; % кількість правил.
    mask = zeros(rulecount,1); % побітова маска ітератора.
    j=0;
    rulemap = 0; % вектор правил.
    mapp = 1; % ітератор вектора правил.
        while(bitshift(i,-j))
            mask(j+1)= mod(bitshift(i,-j),2);
            j=j+1;
            if(mask(j)) % запам'ятовуємо номера правил.
                rulemap(mapp) = j;
                mapp = mapp+1;
            end
        end
        rulenum = sum(mask);
        % Тут проведіть всі необхідні обрахунки...
end
```

Аналогічно до наведених вище програм можна розробити сценарії для жадібного алгоритму, який полягає у почерговому додаванні найкращих правил. Алгоритм стартує з порожньої бази знань і на першій ітерації додає в неї одне правило, яке забезпечує мінімальну нев'язку. 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 з рис. 25 наведено на рис. 29. Найкращою

виявилась база знань з трьох нечітких правил, з порядковими номерами 1, 2 та 4. В звіті на кривих навчання слід виділити найточнішу базу знань та вказати, з яких правил вона складається.

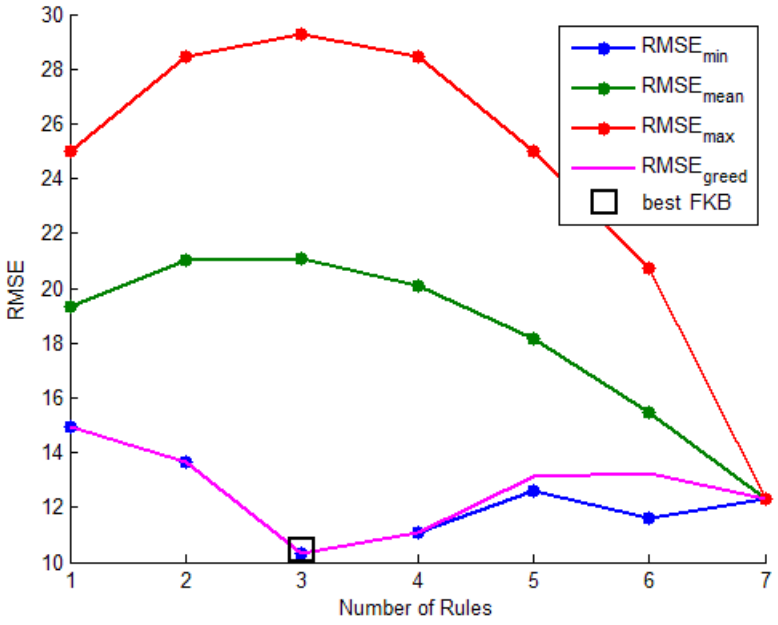


Рисунок 29 – Криві навчання бази знань Мамдані з рис. 25

Для відображення в звіті функцій належності входів та виходу (рис. 30) можна викликати функцією **`plotmf`**.

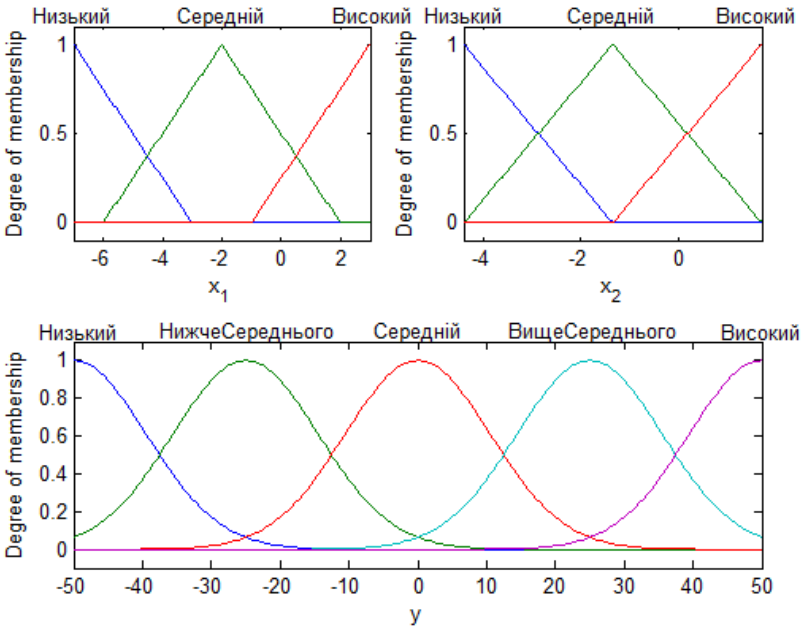


Рисунок 30 – Графіки функцій належності входів бази знань Мамдані

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 масива дублюючих елементів;

min – значення та порядковий номер мінімального елементу масива;

max – значення та порядковий номер максимального елементу масива;

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. Зважаючи на обчислювальну складність, скільки правил доцільно розглядати повним перебором і у яких випадках?
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?

4. Побудова кривої навчання нейронної мережі

Мета – експериментально встановити залежність якості нейронної мережі від обсягу навчальної вибірки.

Теоретичні положення

Штучна нейронна мережа – математична модель, яка представляє собою систему з'єднаних між собою простих обчислювальних елементів (нейронів). Ця математична модель імітує функціонування біологічної нейронної мережі. Нейрон складається з тіла, синапсів – утворень на дендритах, які отримують вхідний сигнал, та аксону – відросту, по якому передається вихідний сигнал (рис. (16)31). Основною властивістю нейронів є здатність до збудження та пригнічення під впливом вхідних електричних сигналів, що надходять по синапсам.

Існує багато типів штучних мереж: мережі прямого розповсюдження; рекурентні нейронні мережі; самоорганізаційні карти тощо. В лабораторній роботі досліджується багатошаровий персептрон – нейронна мережа прямого розповсюдження. В багатошаровому персептроні кожен нейрон представляється досить простою структурою (див. рис. (16)31). Нейрон містить суматор, на який подаються зважені вхідні сигнали. Після суматора сигнал подається на функцію активації, яка відображає його у вихідний сигнал. Кожен вхід нейрона має вагу, який послаблює чи підсилює вхідний сигнал. Вагові коефіцієнти можуть бути як додатними, так і від’ємними, тобто вхідний сигнал може як збуджувати нейрон, так і пригнічувати його. Відтак кожен нейрон мережі функціонує таким чином:

$$S_j = \sum_{i=\overline{1,m}} w_{ij} a_i + w_{0j}$$
$$a_j = \mathrm{f}_j(S_j), \qquad (16)$$

де S_j – зважена сума вхідних сигналів j-го нейрону;

w_{ij} – вага i-го входу j-го нейрону, $i = \overline{1,m}$;

a_i – значення i-го входу j-го нейрону, $i = \overline{1,m}$;

m – кількість входів j-го нейрону;

w_{0j} – зміщення j-го нейрону;

a_j – вихід j-го нейрону;

$\mathrm{f}_j(.)$ – функція активації j-го нейрону.

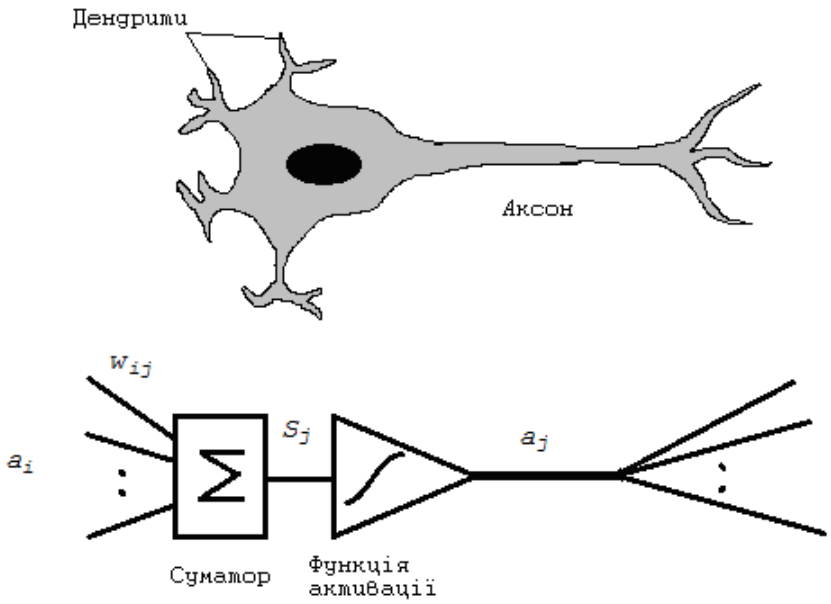


Рисунок 31 – Структура нейрона

Зміщення w_{0j} відіграє важливу роль у активізації нейрона, адже дозволяє активувати нейрон за нульових значень вхідних сигналів. Це дозволяє у простий спосіб реалізувати логічні схеми на базі нейронів (рис. (17)32). Для зручності в формулі (16) зміщення вносять під знак суми,

додавши вхід x_0, який завжди має одиничне значення. Тоді S_j запишемо так:

$$S_j = \sum_{i=0,\,m} w_{ij} a_i \,. \tag{17}$$

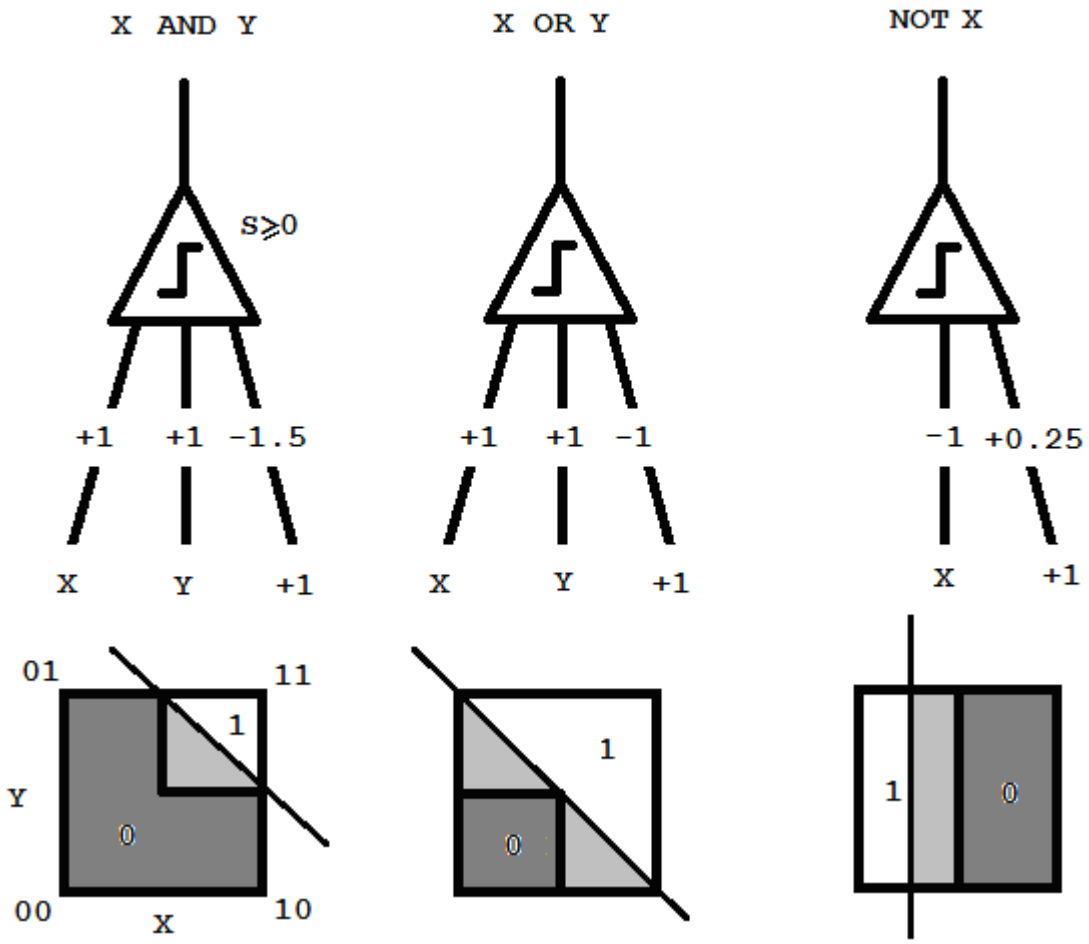


Рисунок 32 – Логічні елементи на базі штучних нейронів [26]

На базі логічних елементів з рис. 32 можна створити нейрокомп'ютер, який не програмується за звичайною фон-неймановською концепцією, а навчається. Замість створення схем послідовностей різних логічних операцій, операція нейрона визначається вхідними вагами, а отже для програмування алгоритму достатньо налаштувати ваги. Навчання – це процес налаштовування ваг вхідних зв'язків нейронів штучної нейронної мережі.

Функція активації $f(S_j)$ може бути реалізована пороговою, лінійною, сигмоїдною функціями або гіперболічним тангенсом. Головними властивостями, яким повинна відповідати функції активації, є існування похідної у кожній її точці та простота обрахунку власне похідної. Тому порогова функція не використовується в сучасних методах навчання багатошарових нейронних мереж. Найчастіше застосовується сигмоїдна функція активації, яка записується таким чином:

$$f(S_j) = \frac{1}{1+e^{-kS_j}}, \tag{18}$$

де k – коефіцієнт крутизни.

Сигмоїдна функція на всій області визначення є неперервною та гладкою. Нею можна апроксимувати всі вище згадані функції активації (рис 33). Також сигмоїдна функція має дуже просту формулу похідної:

$$\frac{df(S_j)}{dS_j} = k \cdot f(S_j)(1 - f(S_j)) = ka_j(1 - a_j).$$

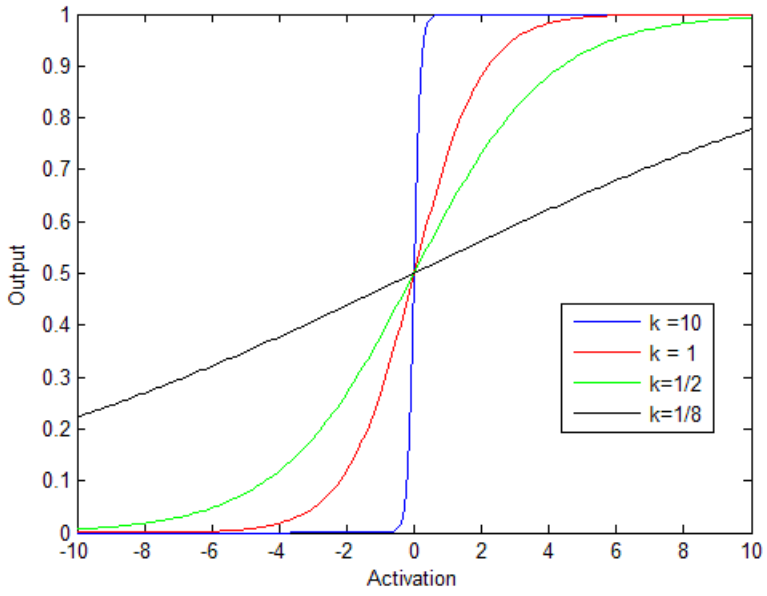


Рисунок 33 – Сигмоїдна функція активації нейрону

Одношаровий персептрон (рис. (17)32) можна розглядати як лінійний дискримінатор. Рівняння прямої, яке задає границю розділу класів, є таким: $w_1x_1 + w_2x_2 + w_0 \geq 0$. Застосування одношарового персептрона

обмежене класом лінійно роздільних задач. Але поєднавши послідовно виходи та входи шарів, які складаються з простих нейронів, можна отримати довільну границю розділу класів. Наприклад, нейронна мережа з рис. 34 описує складний логічний елемент XOR – “виключне або”. Як видно з рис. 34, вихідний елемент реалізує операцію AND, що дозволяє виділити не замкнуту область між прямими, що описуються рівняннями нейронів попереднього шару. Додавши один нейрон на прихованому шарі, можна отримати трикутну замкнуту область. Збільшуючи таким чином кількість нейронів в прихованому шарі можна описати будь-який опуклий багатокутник. А додавши ще один проміжний прихований шар можна описати довільний багатокутник.

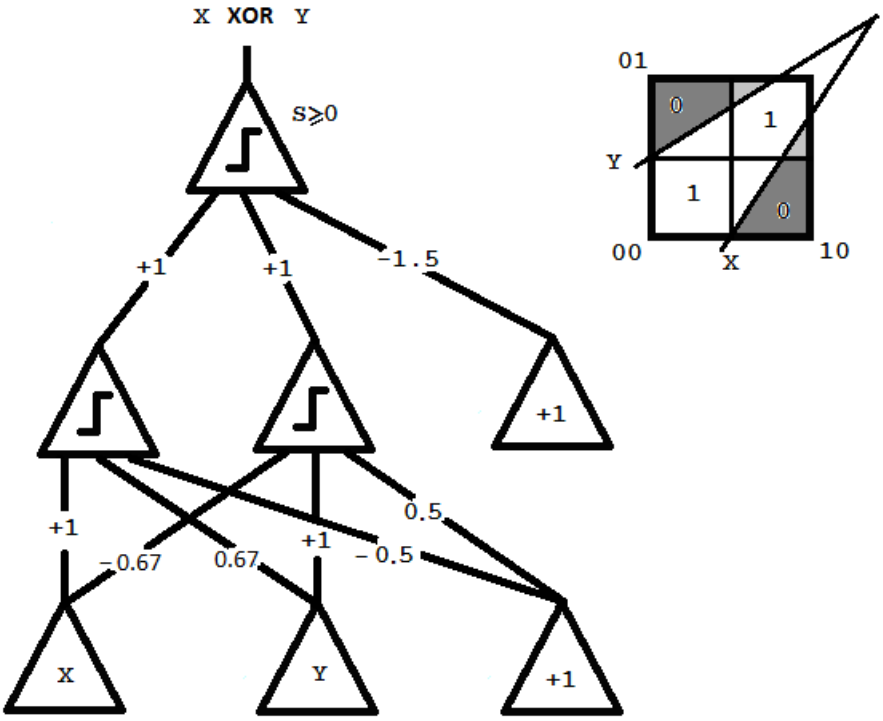


Рисунок 34 – Логічна операція XOR на базі штучних нейронів

Теоретично доведено, що двошарова нейрона мережа (рис. 35) з нелінійною функцією активації є універсальним апроксиматором, який здатен описати довільну гладку залежність з якою завгодно точністю. Але залишається відкритим питання як отримати нейрону мережу з потрібною точністю – скільки потрібно нейронів у прихованому шарі, якими мають бути їх параметри?

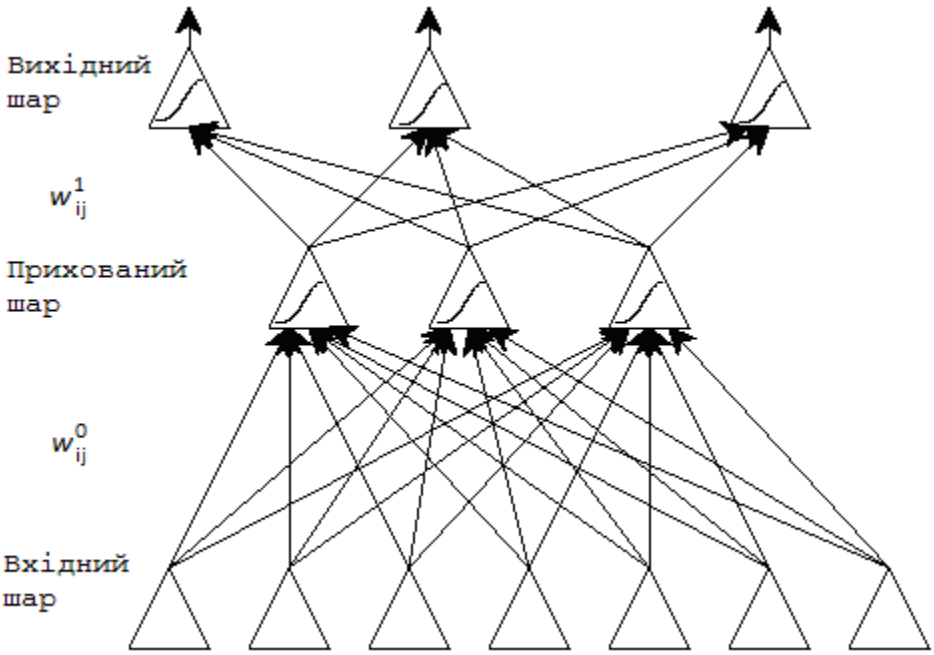


Рисунок 35 – Структура двошарового персептрона [27]

Встановлення невідомих параметрів нейронної мережі здійснюється під час її *навчання*. Навчання проводиться таким чином: мережі надається приклад задачі, який вона вирішує. Якщо відповідь мережі відрізняється від бажаної, тоді параметри мережі змінюють, таким чином щоб її вихід наблизився до очікуваного.

Закріплення результатів проводиться шляхом повторення прикладів, доки мережа не почне видавати рішення з бажаною точністю. Якщо бажана точність не досягається після численних ітерацій навчання, це може свідчити про недостатню складність мережі. Наприклад, скільки би одношаровий персептрон не навчали на прикладах задачі XOR, цю логічну функцію він не реалізує. Під час ідентифікації прикладних залежностей необхідна складність нейронної мережі наперед невідома, і не може бути достовірно визначена за формальними ознаками. Раціональним є підхід до поступового нарощування складності мережі. Тобто спочатку залежність слід ідентифікувати за допомогою простих моделей, і у разі незадовільної точності перейти до складних моделей.

Навчаючи мережу з великою кількістю нейронів протягом тривалого часу слід пам’ятати про можливість її перенавчання. В результаті настроювання мережі на навчальній вибірці можна отримати нев’язку близьку до нуля. В цьому випадку мережа зазвичай втрачає генералізуючі

властивості – вона добре апроксимує дані з навчальної вибірки, але видає велику нев'язку на нових, невідомих реальних даних. Якщо ідентифікація здійснена вдало, тоді модель має приблизно однакову нев'язку як на даних, що використовувалися для навчання, так і для нових даних (рис. 36а). У випадку перенавчання нев'язка для нових даних буде велика (рис. 36б).

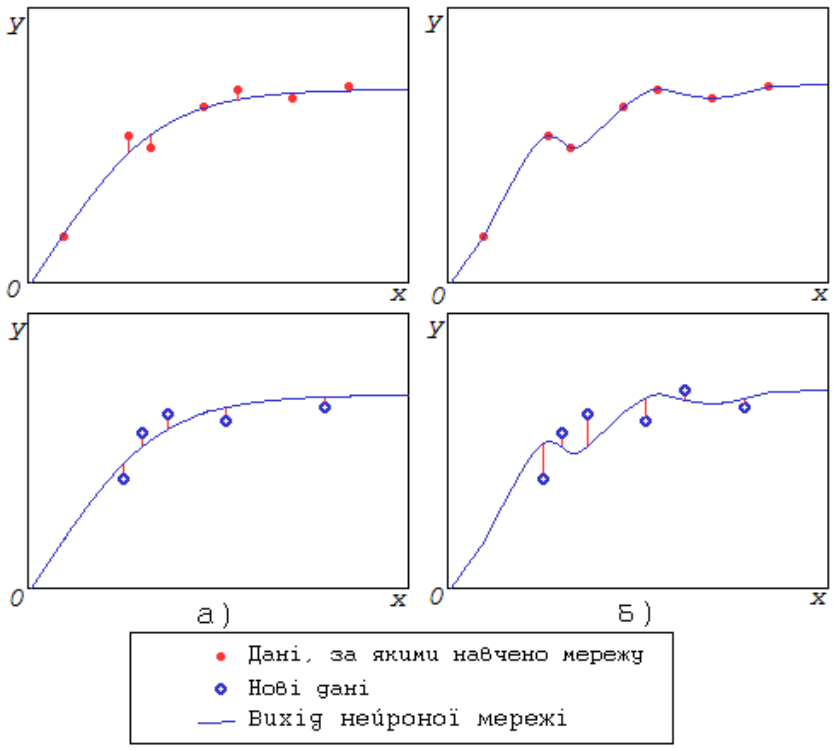


Рисунок 36 – Ідентифікація залежності нейронною мережею:
а) адекватне навчання; б) перенавчання.

Для запобігання перенавчання часто застосовують тактику ранньої зупинки. Для цього дані розбивають на навчальну та тестову вибірки. За навчальною вибіркою настроюють параметри нейронної мережі. Після кожної ітерації навчання точність ідентифікації перевіряють на тестовій вибірці. Типова залежність нев'язки від тривалості навчання нейронної мережі наведена на рис. 37. Вона показує, що на тестовій вибірці нев'язка спочатку зменшується, а потім починає зростати. Оптимальною буде нейронна мережа з мінімальною нев'язкою на тестовій вибірці. Таким чином, під час навчання використовується навчальна вибірка – для зміни параметрів нейронної мережі, і тестова вибірка – для визначення моменту припинення навчання.

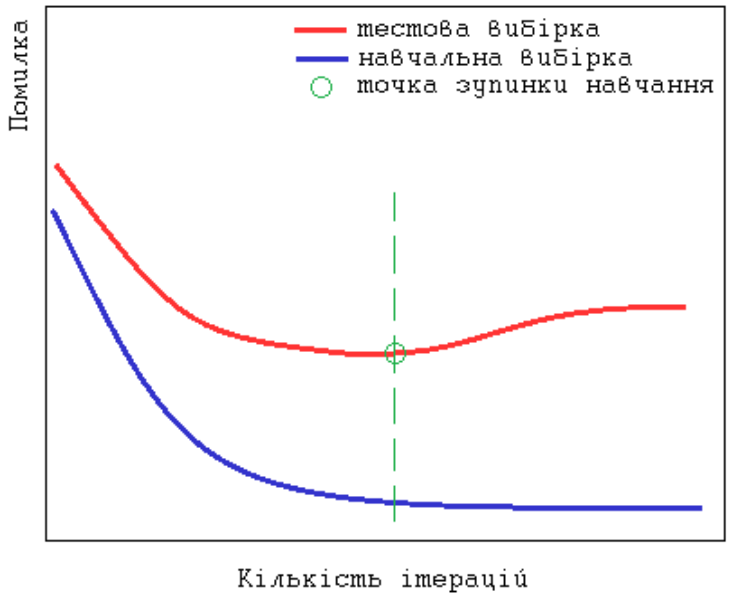


Рисунок 37 – Динаміка навчання нейронної мережі

Особливістю нейронних мереж є прихованість знань. Знаючи архітектуру мережі і її параметри ми не можемо якісно описати залежність між її входами та виходами. Щоб мати уяву про цю залежність потрібно сформувати вибірку даних, подати їх на вхід мережі, розрахувати значення на виході і за результатами побудувати відповідні графіки. Зауважимо, що якщо за вхідного сигналу $x=1$ на виході мережі отримано значення $y=10$, а для $x=2$ $y=20$, то немає ніякої гарантії, що за вхідного сигналу $x=1.5$ значення на виході буде в діапазоні $y\in[10,20]$. На противагу нейронним мережам, для регресійної залежності типу $y=5x_1-x_2+3$ можна без будь-яких розрахунків сказати, що залежність між входами і виходом є лінійною, по x_1 вона монотонно зростаюча, а по x_2 – монотонно спадна. При цьому чутливість виходу по входу x_1 в 5 разів вища, ніж по x_2.

Оскільки під час налаштування параметрів нейронної мережі використовується і навчальна і тестова вибірки, тому для перевірки моделі потрібні додаткові дані. Для цього використовується контрольна вибірка, за якою перевіряють генералізуючу властивість нейронної мережі, тобто її точність поза точками навчання. Обсяг контрольної вибірки має бути великим, особливо у випадку складних нейронних мереж.

Дельта-правило. Розглянемо двошаровий персептрон (див. рис. 35). Мережа має вхідний шар, який приймає вхідні сигнали та передає їх по зв'язкам w_{ij} на суматори нейронів. Якщо вагові коефіцієнти задати випадково, тоді значення на виході мережі буде значно відрізняється від бажаного. Зміна ваг спричинить відхилення вихідної величини, її наближення чи навпаки віддалення від бажаного значення. Це відхилення можна чисельно оцінити функцією помилки. Змінюючи вектор параметрів w_{ij}, додаючи Δw_{ij} у напрямку найшвидшого зменшення помилки можна якнайближче наблизитись до бажаного значення. Напрямок найшвидшого зростання функції називається *градієнтом*, і визначається вектором частинних похідних функції помилки. Розрахувавши градієнт функції помилки можна отримати Δw_{ij}. Описаний вище принцип навчання нейронної мережі шляхом налаштування її параметрів покладено в основу градієнтного методу навчання.

Функція помилки задається таким чином:

$$E_p = \frac{1}{2} \sum_{j=\overline{1,n}} (t_j - a_j)^2, \tag{19}$$

де E_p – помилка мережі для p-го обєкту навчальної вибірки;

t_j – бажаний вихід j-го нейрону вихідного шару;

a_j – дійсний вихід j-го нейрону вихідного шару.

Градієнт Δw_{ij} – це вектор частинних похідних $\frac{\partial E_p}{\partial w_{ij}}$. За функції активації $f(S_j)$ градієнт Δw_{ij} розраховується таким чином:

$$\frac{\partial E_p}{\partial w_{ij}} = \frac{\partial E_p}{\partial a_j} \frac{\partial a_j}{\partial S_j} \frac{\partial S_j}{\partial w_{ij}};$$

$$\frac{\partial E_p}{\partial a_j} = 2 \cdot \frac{1}{2} \sum_j (t_j - a_j) \cdot -1 = -(t_j - a_j);$$

$$\frac{\partial a_j}{\partial S_j} = f(S_j)';$$

$$\frac{\partial S_j}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_i (w_{ij} x_i) = x_i;$$

$$\frac{\partial E_p}{\partial w_{ij}} = -(t_j - a_j) \cdot f(S_j)' \cdot x_i .$$

Оскільки градієнт вказує напрямок зростання функції, зміщення вектору w_{ij} проводиться у напрямку антиградієнту. Для цього градієнт множимо на -1. Швидкість навчання визначається довжиною кроку ε. Узагальнену формулу градієнту запишемо так:

$$\Delta w_{ij} = -\varepsilon \frac{\partial E}{\partial w_{ij}} = \varepsilon \cdot \delta_j \cdot x_i;$$
$$\delta_j = (t_j - a_j) \cdot f(S_j)'. \qquad (20)$$

Тепер здійснимо крок від точки параметрів w_{ij} на відстань Δw_{ij} у напрямку найшвидшого зменшення помилки:

$$w_{ij}(\tau + 1) = w_{ij}(\tau) + \Delta w_{ij}(\tau), \qquad (21)$$

де $w_{ij}(\tau + 1)$ – нове значення ваг на ітерації $\tau + 1$;

$w_{ij}(\tau)$ – поточне значення ваг на ітерації τ.

Нові ваги персептрона повинні наблизити вихід мережі до бажаного, цим самим зменшивши помилку. Процедуру (21) повторюємо, поки не буде досягнута бажана точність. Але частіше критерієм зупинки процесу є обмеження на кількість ітерацій або на тривалість навчання.

Зворотне розповсюдження помилки. Розрахувати дельту δ для вихідного шару досить просто, знаючи бажаний вихід. Але цільове значення виходів прихованого шару є невідомими, тому застосувати дельта правило, на перший вигляд, неможливо:

$$w_{ij}^l(\tau + 1) = w_{ij}^l(\tau) + \varepsilon \cdot \delta_j^l \cdot x_i^l .$$

Але розглянувши детально похідну прихованого шару $\frac{\partial E_p}{\partial w_{ij}^{l-1}}$ можна легко пересвідчитись, що форма розрахунку зберігається, різниця тільки в

тому, як обрахувати δ. Розглянемо похідну $\frac{\partial E_p}{\partial w_{ij}^{l-1}}$, де $x_j = a_j^{l-1}$ є виходом прихованого шару від відповідної суми S_j^{l-1}:

$$\frac{\partial E_p}{\partial w_{ij}^{l-1}} = \frac{\partial E_p}{\partial a_k}\frac{\partial a_k}{\partial S_k}\frac{\partial S_k}{\partial x_j}\frac{\partial a_j^{l-1}}{\partial S_j^{l-1}}\frac{\partial S_j^{l-1}}{\partial w_{ij}^{l-1}} = \frac{\partial E_p}{\partial S_j^{l-1}}\frac{\partial S_j^{l-1}}{\partial w_{ij}^{l-1}} = \frac{\partial E_p}{\partial S_j^{l-1}} x_i^{l-1}.$$

Відповідно, формулу градієнтного спуску можна записати так:

$$w_{ij}^l(\tau+1) = w_{ij}^l(\tau) - \varepsilon\frac{\partial E_p}{\partial S_j^l} x_i^l.$$

Розрахуємо дельту δ попереднього шару w_{ij}^{l-1}, вважаючи w_{jk}^l вихідним шаром, значення дельти δ якого відоме:

$$\frac{\partial E_p}{\partial S_j^{l-1}} = \sum_k \frac{\partial E_p}{\partial a_k}\frac{\partial a_k}{\partial S_k}\frac{\partial S_k}{\partial x_j}\frac{\partial a_j^{l-1}}{\partial S_j^{l-1}} = \left|\begin{array}{l} \frac{\partial E_p}{\partial a_k}\frac{\partial a_k}{\partial S_k} = \delta_k^l \\ \frac{\partial S_k}{\partial x_j} = w_{jk}^l \\ \frac{\partial a_j^{l-1}}{\partial S_j^{l-1}} = f_j^{l-1}(S_j^{l-1})' \end{array}\right. = \sum_k \delta_k^l w_{jk}^l f_j^{l-1}(S_j^{l-1})'.$$

Звідси отримаємо узагальнену формулу обрахунку:

$$w_{ij}(\tau+1) = w_{ij}(\tau) + \Delta w_{ij}(\tau),$$

$$\Delta w_{ij}(\tau) = \varepsilon \cdot \delta_j \cdot x_i,$$

де $\delta_j = (t_j - a_j)\cdot f(S_j)'$ – для вихідного шару;

$\delta_j = f(S_j)' \sum_{k=\overline{1,n}} \delta_k^{l+1} w_{jk}^{l+1}$ – для прихованого шару.

Як бачимо, попереднє значення δ, залежить від наступного, тому кажуть що помилка розповсюджується у зворотному напрямку від виходу до входу. Цей метод навчання отримав назву – *алгоритм зворотного розповсюдження помилки.*

Завдання на роботу

В лабораторній роботі розглядається ідентифікації залежності типу "2 входи – 1 вихід" за допомогою нейронної мережі. В лабораторній роботі потрібно побудувати криву навчання нейронної мережі у вигляді залежності середньо-квадратичної нев'язки (*RMSE*) на контрольній вибірці від обсягу навчальної вибірки. Для цього необхідно виконати 4 завдання.

1. Побудувати трьохвимірний графік залежності з табл. 12, ідентифікацію якої потрібно здійснити за допомогою нейронної мережі.

2. Згенерувати дані для навчання та перевірки нейронної мережі з заданої аналітичної залежності. Експерименти провести для навчальних вибірок з 10, 20, 30, 40, 50, 75 та 100 пар "входи – вихід". Кожна навчальна вибірка меншого розміру має бути підмножиною кожної навчальної вибірки більшого розміру. Обсяг тестової та контрольної вибірок – 100 та 1000 пар "входи – вихід". Дані з навчальної, тестової та контрольної вибірок мають бути різними.

3. Провести дослідження для нейронної мережі "багатошаровий персептрон" (Feed-forward backprop) з лінійною функцією активації (pureline) вихідного нейрона. Кількість нейронів на прихованому шарі та тип їх функції активації обрати з табл. 12 згідно до варіанту.

4. Побудувати графіки регресії "вихід – вихід" та поверхні помилки на контрольній вибірці для трьох навчених мереж: найгіршої, посередньої та найкращої.

Таблиця 12 – Варіанти завдань

Варіант	Аналітична залежність	Прихований шар		Діапазон	
		Нейронів	Функція активації	x_1	x_2
1	$y = x_1\sqrt{x_2}$	3	logsig	[2, 22]	[2, 14]
2	$y = \frac{{x_1}^2 + {x_2}^3}{x_1 + x_2}$	2	tansig	[0, 20]	[6, 10]
3	$y = \frac{x_1 + x_1x_2 - \sqrt{x_2}}{x_1 + x_2}$	3	logsig	[1, 5]	[0.5, 2]
4	$y = \frac{x_1 + x_2}{\sqrt{x_1} + {x_2}^2 - 1}$	4	tansig	[0.6, 2]	[1, 3]

Варіант	Аналітична залежність	Прихований шар		Діапазон	
		Нейронів	Функція активації	x_1	x_2
5	$y = \frac{{x_1}^3 \sin(0.2x_2)}{x_1 + {x_2}^2}$	3	logsig	[100, 150]	[50, 100]
6	$y = (1+\sin(x_1)^2)^{x_2}$	4	tansig	[0, 3]	[0.5, 2]
7	$y = \frac{\sin x_1}{x_2}$	2	logsig	[0, 5]	[1, 7]
8	$y = \frac{{x_1}^2 - {x_2}^3}{2x_1 + x_2}$	3	tansig	[0, 4]	[2, 6]
9	$y = \frac{x_1 - \sqrt{x_2}}{\sqrt{x_1} + x_2}$	4	logsig	[0, 0.5]	[0.05, 0.5]
10	$y = \frac{x_1 - {x_2}^3}{x_1 x_2}$	5	tansig	[10, 25]	[1, 7]
11	$y = (x_1 - 4) \cdot (x_2 - 5)^3$	3	logsig	[20,80]	[2, 7]
12	$y = 7\sqrt{x_1} - 6x_1x_2$	4	tansig	[30, 50]	[50, 100]
13	$y = {x_1}^4 \sin(x_2) - \sqrt{x_2}$	2	logsig	[3, 6]	[5, 10]
14	$y = {x_1}^2 - {x_2}^3 tg(0.1x_1)$	3	tansig	[-2, 3]	[1, 5]
15	$y = \frac{x_1 + {x_2}^3}{2\sqrt{x_1 x_2}}$	4	logsig	[2, 7]	[10, 20]
16	$y = 0.01{x_1}^2 tg(x_2)$	5	tansig	[-20, 25]	[-1, 1.3]
17	$y = (x_1 + x_2)\sin(x_1 + x_2)$	3	logsig	[1, 4]	[0,5, 2.5]
18	$y = \frac{{x_1}^2 + x_2}{e^{x_1 x_2}}$	4	tansig	[-4, -3]	[0, 1]
19	$y = \sin(x_1)\cos(2x_2)$	3	logsig	[1, 2]	[1, 2]
20	$y = \frac{\sqrt{x_1^2 + x_2^2}}{x_1 + x_2}$	4	tansig	[2, 4]	[3, 5]
21	$y = \frac{x_1^2 + \sin(0.1x_2)}{\cos x_1 + x_2}$	3	logsig	[1, 5]	[3, 7]
22	$y = \frac{x_1^2 + x_2}{0.1e^{x_1} + x_2}$	4	tansig	[-5, 5]	[3, 7]

Варіант	Аналітична залежність	Прихований шар		Діапазон	
		Нейронів	Функція активації	x_1	x_2
23	$y=(x_1x_2)\sqrt{x_1^2+x_2^2}$	5	logsig	[5, 20]	[-10, 10]
24	$y=(x_1-x_2)(x_1^2+x_1x_2+x_2^2)$	4	tansig	[0.1, 2]	[0, 1.5]

Рекомендації до виконання

Для поглибленого вивчення теоретичного матеріалу рекомендуємо літературу [5, 8, 9, 13, 15, 16, 24, 26, 27]. Нейронні мережі можна проектувати в різних пакетах, серед яких виділимо Neural Network Toolbox в системі MATLAB 7 та пакет Statistica. Доцільно ознайомитись з описом GUI-модулів пакету Neural Network Toolbox з [24].

Під час виконання *першого завдання* доцільно застосувати функції **surf** або **mesh**. Приклад програми побудови графіка функції $y=x_1\sqrt{x_2}$ в області $x_1\in[2,22]$ та $x_2\in[2,14]$ наведено нижче. Результати роботи програми показані на рис 38.

```
%Побудова графіка функції y=a*b^0.5
%в області x1 є[2,22] та x2 є[2,14].
fun = @(x1, x2) x1.*(x2.^0.5);
n=10;
x1=linspace(2, 22, n);
x2=linspace(2, 14, n);
INP=zeros(n*n,2);
OUT=zeros(n*n,1);
k=1;
for i=1:n
    for j=1:n
        INP(k,1)=x1(i);
        INP(k,2)=x2(j);
        OUT(k)=fun(x1(i), x2(j));
        k=k+1;
    end
end
y=reshape(OUT,n,n)
surf(x1,x2,y);
xlabel('x_1');
ylabel('x_2');
zlabel('y');
```

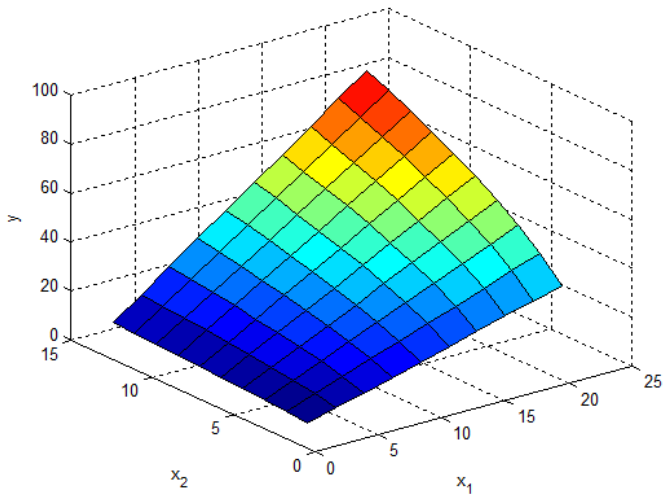


Рисунок 38 – Поверхня залежності $y = x_1\sqrt{x_2}$

Для виконання *другого завдання* спочатку випадковим чином згенеруйте 1200 вхідних значень та за аналітичною залежністю розрахуйте вихідне значення. Включіть 100 даних у навчальну вибірку, 100 даних – в тестову та 1000 – в контрольну. Збережіть вибірки даних в файл.

Для виконання *третього завдання* можна використати GUI-модуль nntool (рис. 39). Модуль nntool має такі особливості:

- вхідні та вихідні значення в вибірках даних потрібно описати в робочій області різними змінними;
- кожному об’єкту відповідає один стовпець матриці даних (в інших пакетах об’єкту відповідає рядок матриці даних). Для транспонування матриці використовуйте операцію ';
- входи нейронній мережі не вважаються окремим шаром, тобто шар №1 – це перший прихований шар.

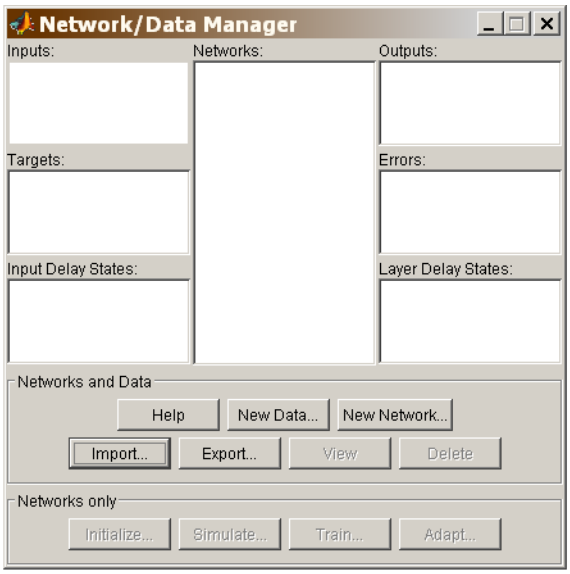


Рисунок 39 – Головне вікно GUI-модуля nntool

Спочатку слід завантажити вибірки даних в nntool, натиснувши кнопку Import.... Далі в новому вікні (рис. 40) обрати ідентифікатор вхідних даних вибірки, встановити опцію Inputs та натиснути кнопку Import. Аналогічно завантажити вихідні дані вибірки, при цьому перед натиском кнопки Import не забудьте встановити опцію Targets. Повторіть вказані дії для завантаження тестової та контрольної вибірок даних.

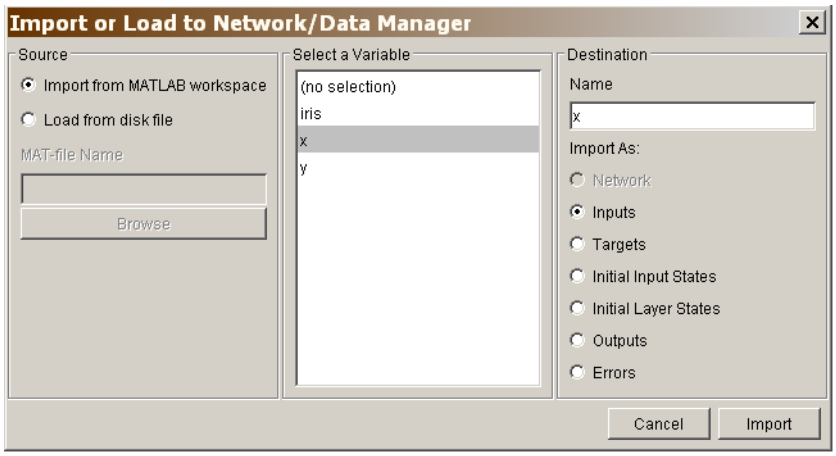


Рисунок 40 – Вікно завантаження даних GUI-модуля nntool

Тепер згенеруйте нейрону мережу натиском кнопки New Network... (див. рис. 39). Відкриється нове вікно (рис. 41), в якому оберіть тип та архітектуру нейронної мережі.

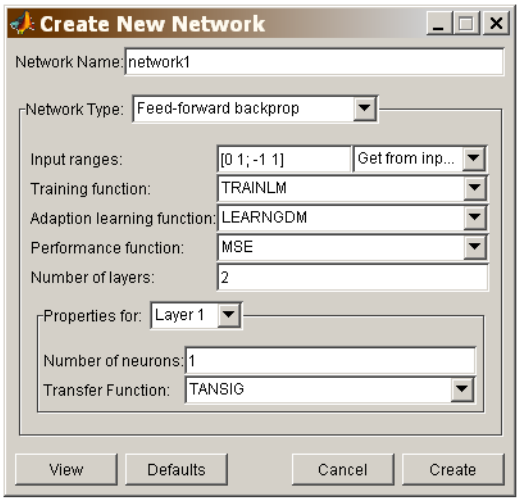


Рисунок 41 – Вікно вибору нейронної мережі в GUI-модулі nntool

Для навчання нейронної мережі натисніть кнопку Train (див. рис. 39). Відкриється нове вікно (рис. 42), в якому оберіть входи та виходи навчальної вибірки (поля Inputs та Targets). Для призначення тестової та контрольної вибірок оберіть закладку Optimal Info та в новому вікні (рис. 43) заповніть поля Validation Data та Testing Data. Для навчання мережі натисніть кнопку Train Network. Для моделювання за допомогою нейронної мережі натисніть кнопку Simulate... (див. рис. 39). Результати моделювання (Outputs...) а також нейрону мережу у вигляді структури даних можна експортувати в робочу область. Для цього натисніть кнопку Export... (див. рис. 39). Не забудьте записати результати моделювання та нейрону мережу на диск.

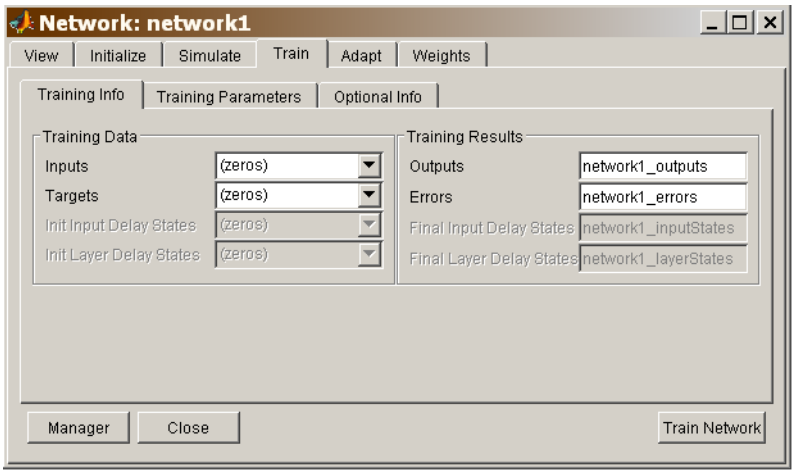


Рисунок 42 – Вікно призначення навчальної вибірки в GUI-модулі nntool

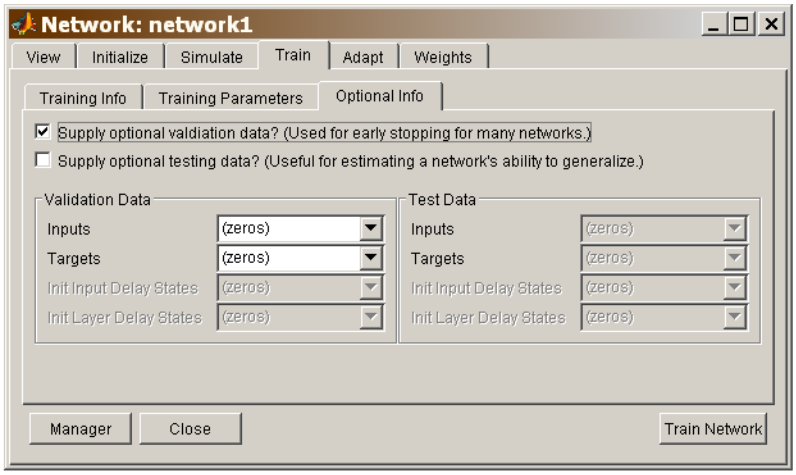


Рисунок 43 – Вікно призначення тестової та контрольної вибірок в GUI-модулі nntool

Якщо у вашій версії MATLAB доступна функція **newff**, створити нейрону мережу можна таким чином:

```
net = newff(INP', OUT', L, Af, learnFunc),
```

де **INP'** та **OUT'** – відповідно вхідна і вихідна змінні вибірки, за якими визначаються кількості входів та виходів мережі;

L – кількість нейронів в прихованих шарах;

Af – список функцій активацій на відповідному шарі;

learnFunc – функція навчання.

net – новостворена мережа

Приклад використання функція **newff**:

```
net = newff(INP', OUT', 3, {'logsig' 'purelin'},'trainlm') .
```

Мережу доцільно навчити за допомогою функції **train**, викликавши її таким чином:

```
[tnet tr] = train(net, INP',OUT') ,
```

де **net** – початкова мережа що піддається навчанню

INP та **OUT** – відповідно вхідні і вихідні дані з навчальної, тестової та контрольної вибірок;

tnet – мережа після навчання;

tr – структура, що містить інформацію про процес навчання.

Щоб вказати межі навчальної, тестової та контрольної вибірок необхідно модифікувати структуру **net** таким чином:

```
net.divideFcn = 'divideind';

net.divideParam = struct('trainInd',A,'valInd',B,'testInd',C),
```

де **A, B** та **C** – індекси навчальної, тестової та контрольної вибірок.

Також можна скористатись функцією **nntrain**, яка здійснює об'єднання масивів і підрахунок індексів (дані передаються як рядки). Лістинг функцією **nntrain** наведено нижче.

```
function [net tr] = nntrain(net, INP, OUT, ...
                                    INP_t, OUT_t, INP_c, OUT_c)
% INP_t, OUT_t - відповідно входи та виходи тестової вибірки.
% INP_c, OUT_c - відповідно входи та виходи контрольної вибірки.
%Визначаємо довжину вибірок:
p_len = size(INP, 1);
vv_len = size(INP_t, 1);
tv_len = size(INP_c, 1);
%Визначаємо довжину загальної вибірки даних:
q = p_len + vv_len + tv_len;
%Визначаємо індекси вибірок після конкатинації:
p_ind = 1:p_len;
vv_ind = (p_len+1):(p_len+vv_len);
tv_ind = (p_len+vv_len+1):q;
%Повідомляємо функцію навчання яким чином ми зберегли навчальну,
%тестову та контрольну вибірки:
net.divideFcn = 'divideind';
net.divideParam =
struct('trainInd',p_ind,'valInd',vv_ind,'testInd',tv_ind);
[net tr] = train(net,[INP; INP_t; INP_c]',[OUT; OUT_t; OUT_c]');
```

Перед кожним навчанням мережі Neural Network Toolbox викликає функції ініціалізації її ваг, присвоюючи їм випадкові значення. Початкові умови значно впливають на результати навчання. Для чистоти експерименту, перед кожним навчанням обнуліть генератор випадкових чисел, це дасть змогу відтворити однакові умови навчання:

```
RandStream.setGlobalStream(RandStream('mrg32k3a','Seed',1234));
```

В пакеті Neural Network Toolbox якість нейронної мережі визначається квадратичною нев'язкою *MSE* на тестовій вибірці, яка зберігається в полі

`tr.tperf`. *MSE* та *RMSE* з формули (2) пов'язані таким чином: $RMSE = \sqrt{MSE}$. Розрахунок *MSE* можна реалізувати таким чином:

```
Y = sim(tnet, INP_c');
E = mse(Y - OUT_c).
```

Для прикладу розглянемо ідентифікацію залежності функції $y = x_1\sqrt{x_2}$ (див. рис. 38) за допомогою нейронної мережі з трьома нейронами в прихованому шарі. На рис. 44 зображено динаміку навчання нейронної мережі. Протягом перших трьох ітерацій мережа швидко навчається, відповідно *MSE* стрімко зменшується. Далі протягом 4 – 6 ітерацій динаміка навчання уповільнюється. На останньому етапу мережа входить в фазу перенавчання, коли вона точно описує ідентифіковану залежність лише в точках навчальної вибірки. Про це свідчить зростання нев'язки на тестових та контрольних вибірках. Перенавчання спричинено замалим обсягом даних для налаштування достатньо складної нейронної мережі.

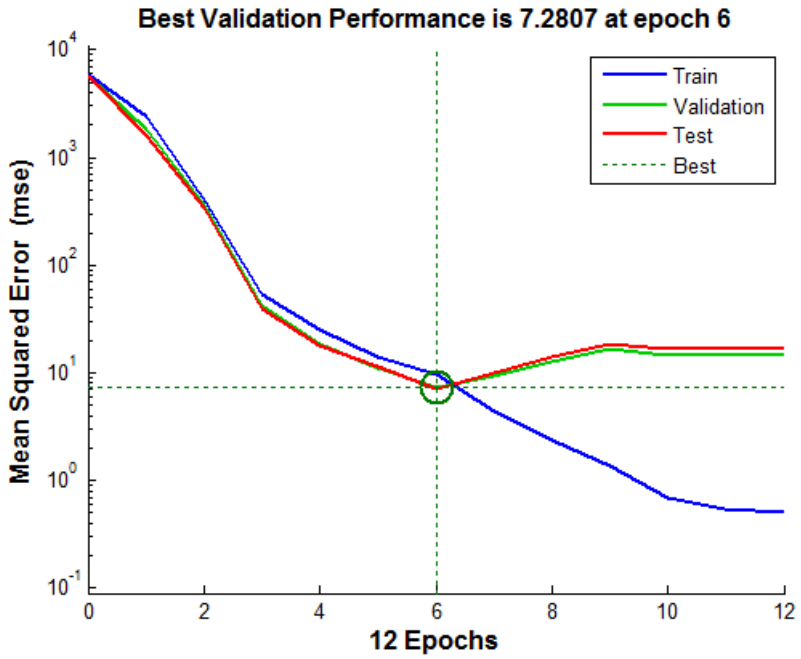


Рисунок 44 – Динаміка навчання нейронної мережі

Для кожної згенерованої вибірки проводимо навчання та фіксуємо якість ідентифікації (рис. 45). За малої кількості прикладів, приблизно до 50, спостерігається значна розбіжність нев'язок на навчальній та контрольній вибірках унаслідок перенавчання. При збільшені кількості

прикладів до 200-320, різниця між нев'язками наближається до 0, тому що велик ий обсяг вибірки запобігає перенавчанню.

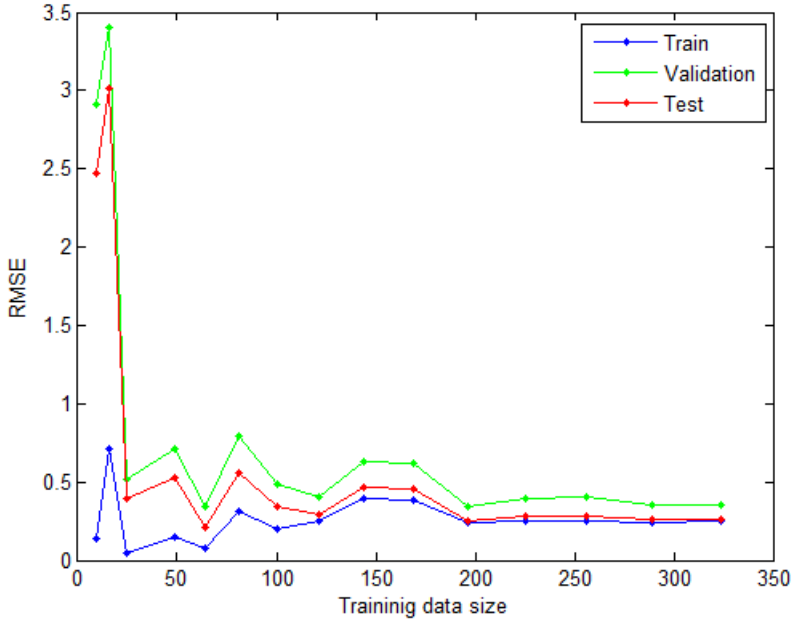
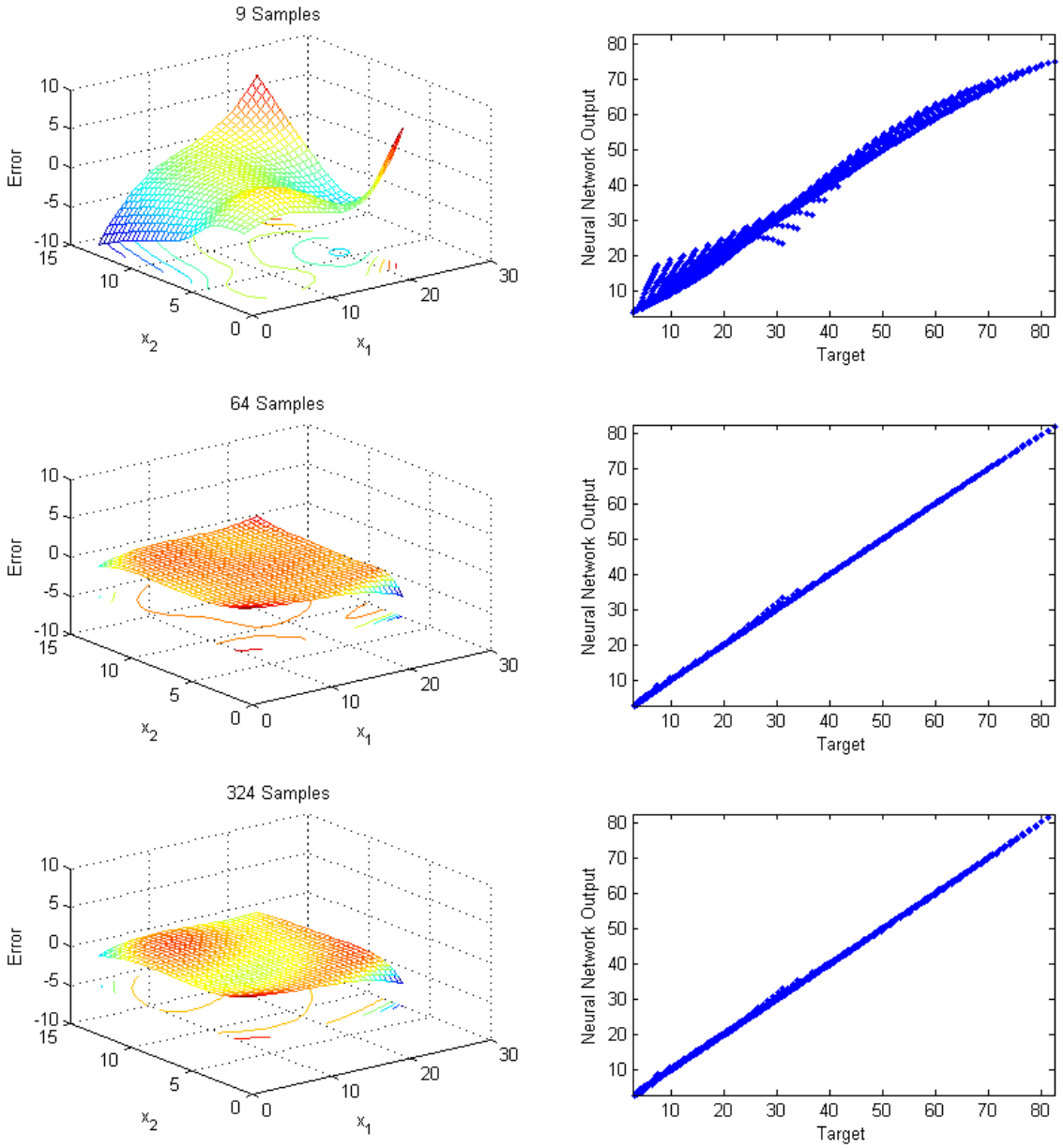


Рисунок 45 – Крива навчання нейронної мережі

Для виконання *4-го завдання* використайте функції **`plot`** та **`mesh`** таким чином:

```
subplot(3,2,1)
Y = sim(nets{1}, INP_c');
mesh(x1, x2, reshape(OUT_c - Y',32,32));
plot(OUT_c, Y);
```

Як приклад на рис. 46 наведено графік експериментально визначеної поверхні помилок нейронної мережі. За короткої навчальної вибірки, нейронна мережа добре апроксимує лише навчальні дані. Чим більше точок ввійшло в навчальну вибірку, тим ближче вихід мережі Y наближається до реальної функції T.



(а) (б)

Рисунок 46 – Тестування нейронної мережі: а) поверхня помилки; б) графік регресії вихідних значень моделі та аналітичної залежності

Під час виконання лабораторної роботи доцільно використовувати такі функції:

`newff` – створення нейронної мережі прямого розповсюдження сигналу;

`nntool` – виклик редактору нейронних мереж;

`sim` – розрахунок вихідних сигналів мережі для поточного вхідного вектору;

`train` – навчання нейронної мережі;

mse – підрахунок квадратичної нев'язки;

surf – побудова поверхонь;

plot3 – побудова трьохвимірних графіків;

reshape – зміна розміру матриці;

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

max – значення та порядковий номер максимального елементу масива;

min – значення та порядковий номер мінімального елементу масива;

minmax – визначення меж значень вектору.

Детальний опис функцій **newff**, **sim** та **train** наведено в **Довіднику ключових функцій**.

Питання для самоконтролю

1. Що таке штучна нейрона мережа?

2. Які особливості дельта-правила навчання нейронної мережі?

3. Які переваги та недоліки алгоритму зворотного розповсюдження помилки?

4. Що таке перенавчання нейронної мережі і як його запобігти?

5. Навіщо потрібна контрольна вибірка?

6. Як впливає кількість даних на якість навчання нейронної мережі?

7. Яка роль зміщення в нейронній мережі?

8. Які функції активації використовуються в нейронних мережах?

9. Як обираються початкові ваги нейронної мережі?

10. Як визначити необхідну кількість прихованих шарів в нейронній мережі?

11. Як визначити необхідну кількість нейронів в прихованого шару?

11. Чи можна застосувати алгоритм зворотного розповсюдження помилки для навчання нейронної мережі з пороговою функцією активації?

12. Що може призвести до того, що після навчання нейронної мережі помилка на тестовій вибірці набагато вище ніж на навчальній?

13. Що може призвести до того, що після навчання нейронної мережі помилки на навчальній і тестовій вибірці майже однакові, але мають досить високе значення?

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.
```

newff

Призначення Створює нову нейронну мережу.

Синтаксис

```
net = newff(INP', OUT', L, Af, learnFunc)
```

Аргументи

INP – матриця вхідних даних навчальної вибірки розміру $m \times n$, де m – кількість спостережуваних випадків та n – кількість входів;

OUT – вектор вихідних значень розміром $m \times k$, де k – кількість виходів;

L – кількість нейронів в прихованих шарах;

Af – список функцій активацій для кожного шару нейронів;

learnFunc – алгоритм навчання.

Вихідні змінні **net** – структура нової нейронної мережі.

Приклад

```
%Створення нейронної мережі з одним прихованим шаром
%з трьома нейронами:
net = newff(INP',OUT',3, {'logsig' 'purelin'},'trainlm')
```

plotfis

Призначення Графічне представлення структури та основних параметрів системи нечіткого виведення.

Синтаксис

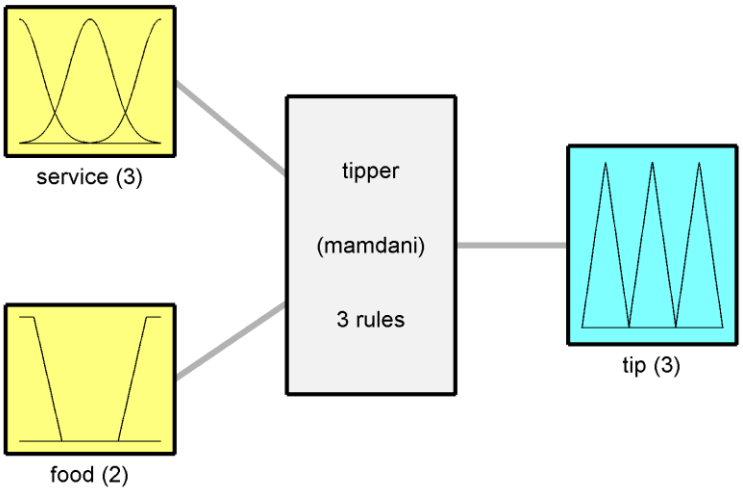
```
plotfis (fis)
```

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).

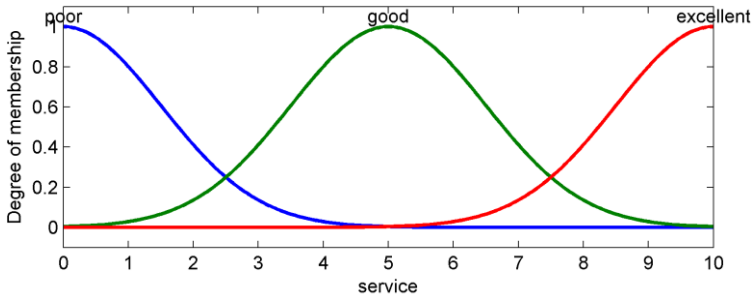
Вихідні змінні
x – матриця абсцис-координат для всіх графіків функцій належності;

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')
```

regress

Призначення: Знаходження коефіцієнтів лінійної регресії для багатовимірної залежності за методом найменших квадратів.

Синтаксис:
```
A = regress(y, X)
[A, Aint, r, rint, stats] = regress(y, X)
[A, Aint, r, rint, stats] = regress(y, X, alpha)
```

Аргументи:
`y` – вектор вихідних значень $m \times 1$, де m – довжина вибірки даних;

`X` – матриця вхідних даних розміру $m \times n$, де n – кількість коефіцієнтів регресійної залежності. Рядки з пропусками значень (`NaN`) під час регресійного аналізу не враховуються;

`alpha` – рівень значущості (значення за замовченням – 0.05);

Рівень значущості використовується для розрахунку меж довірчих інтервалів `Aint` та `rint` з довірчою ймовірністю (1-`alpha`).

Вихідні змінні:
`A` – вектор коефіцієнтів лінійної регресійної моделі `y=X*A`;

`Aint` – довірчі інтервали коефіцієнтів `A`;

`r` – вектор залишків (`y-X*A`);

`rint` – довірчий інтервал залишків `r`. Якщо границі цього інтервал одного знаку, то це ознака викиду даних (промаху вимірювання);

stats – вектор, що містить такі статистики: R^2 – коефіцієнт детермінації; f-критерій Фішера; рівень значущості регресійної моделі та оцінка дисперсії помилок. Статистики розраховуються коректно, якщо **X** містить стовпчик з одиниць, тобто якщо в лінійній регресійній моделі є константа.

Приклад

```
%Завантажуємо дані в робочу область:
load carsmall
x1 = Weight;
x2 = Horsepower;
y = MPG;
%Обраховуємо коефіцієнти лінійної регресійної %моделі з
взаємодією змінних:
X = [ones(size(x1)) x1 x2 x1.*x2];
A = regress(y, X)
```

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 представлення 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 )
```

sim

Призначення: Обчислює вихідні сигнали нейронної мережі.

Синтаксис:
```
ypred = sim(net, INP')
```

Аргументи:
net – нейронна мережа;
INP – матриця вхідних даних навчальної вибірки розміру $m \times n$, де m – кількість спостережуваних випадків та n – кількість входів.

Вихідні змінні:
ypred – матриця вихідних значень нейронної мережі, на вхід якої подано дані **INP**. Розмір матриці $m \times k$, де k – кількість виходів.

Приклад:
```
%Створюємо навчальну вибірку:
X = [0 1 2 3 4 5 6 7 8];
Y = [0 0.84 0.91 0.14 -0.77 -0.96 -0.28 0.66 0.99];
subplot(2,2,1)
plot(X,Y,'o')
legend('Навчальна вибірка')
pause
%Створюємо двошарову мережу з одним прихованим %шаром з
2 нейронів:
net = newff(X,Y,2,{'logsig''purelin'},'trainlm');
%Проводимо навчання:
net = train(net,X,Y);
y2 = sim(net,X)
subplot(2,2,2)
plot(X,Y,'o',X,y2,'*')
legend('Навчальна вибірка','Мережа після навчання')
```

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**. Під час перехресної перевірки випадковим чином формуються підвибірки приблизно однакового розміру. Далі 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 дерева рішень. 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 формулою (9);

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);
```

train

Призначення Навчання нейроної мережі.

Синтаксис `[tnet tr] = train(net, INP',OUT')`

Аргументи **net** – початкова нейронна мережа;

INP – матриця вхідних даних навчальної вибірки розміру $m \times n$, де m – кількість спостережуваних випадків та n – кількість входів;

OUT – матриця вихідних значень розміром $m \times k$, де k – кількість виходів;

Щоб вказати межі навчальної, тестової та контрольної вибірок необхідно модифікувати структуру **net** таким чином:

```
net.divideFcn = 'divideind';
net.divideParam =...
        struct('trainInd',A,'valInd',B,'testInd',C)
```

Вихідні змінні **tnet** – мережа після навчання;

tr – структура, що містить інформацію про процес навчання.

Приклад

```
%Створюємо навчальну вибірку:
X = [0 1 2 3 4 5 6 7 8];
Y = [0 0.84 0.91 0.14 -0.77 -0.96 -0.28 0.66 0.99];
subplot(2,2,1)
plot(X,Y,'o')
legend('Навчальна вибірка')
pause
%Створюємо двошарову мережу з одним прихованим %шаром з
2 нейронів:
net = newff(X,Y,2,{'logsig''purelin'},'trainlm');
y1 = sim(net,X)
subplot(2,2,2)
plot(X,Y,'o',X,y1,'x')
legend('Навчальна вибірка', 'Мережа до навчання')
pause
%Проводимо навчання:
net = train(net,X,Y);
y2 = sim(net,X)
subplot(2,2,3)
plot(X,Y,'o',X,y2,'*')
legend('Навчальна вибірка','Мережа після навчання')
```

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:

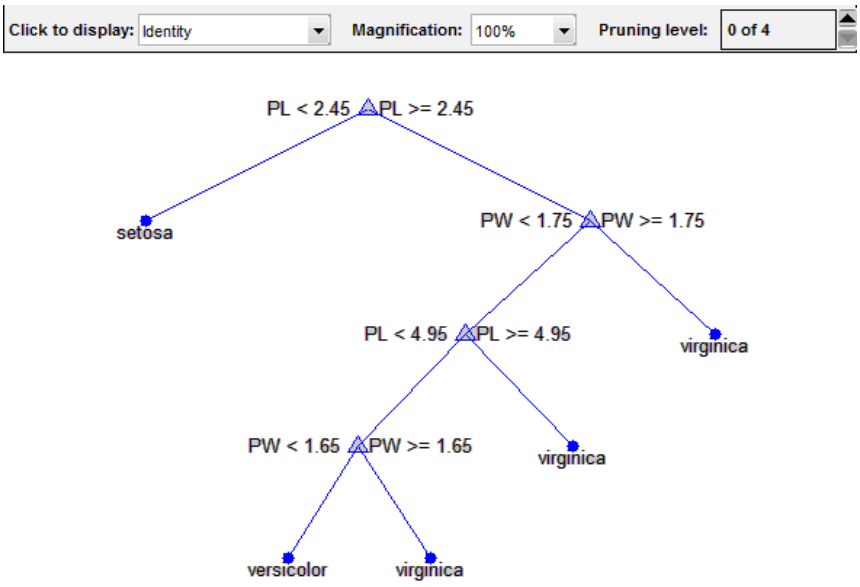
'names' – назви вхідних змінних згідно до їх порядку в навчальній вибірці;

'prunelevel' – кількість рівнів підрізання дерева.

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

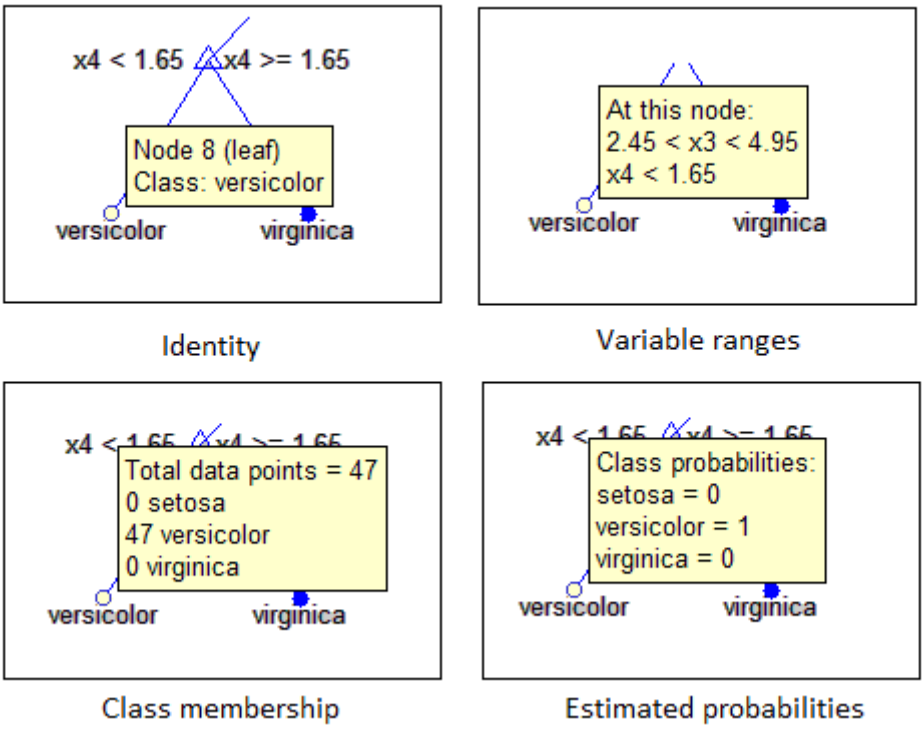
Приклад
```
%Завантажуємо дані задачі класифікації ірисів:
load fisheriris;
%Створюємо дерево рішень:
T = classregtree(meas, species,...
                 'names',{'SL' 'SW' 'PL' 'PW'})
view(T);
```

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 // Математические вопросы кибернетики. – 2004. – Т. 13. – С. 5-36.
3. Городецкий В.И. Методы и алгоритмы коллективного распознавания / Городецкий В.И., Серебряков С.В. // 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. – Режим доступу: http://www.machinelearning.ru/wiki/images/7/7e/Dj2010up.pdf
5. Ежов А.А. Нейрокомпьютинг и его применение в экономике и бизнесе / А.А. Ежов, С.А. Шумский. – М.: 1998. – 222 с.
6. Ивахненко А.Г. Долгосрочное прогнозирование и управление сложными системами / А.Г. Ивахненко. –К.: Техніка, 1975. – 312 с.
7. Ивахненко А.Г. Индуктивный метод самоорганизации моделей сложных систем / А.Г. Ивахненко. – К.: Наукова думка, 1982. – 296 с.
8. Любунь З.М. Основи теорії нейромереж. Текст лекцій / З.М. Любунь. – Львів: Вид. центр ЛНУ імені Івана Франка, 2006. – 140 с.
9. Медведев В.С. Нейронные сети. MatLab 6 / В.С. Медведев, В.Г. Потемкин. – М.: Диалог-МИФИ, 2002. – 496с.
10. Начало работы с MATLAB / Пер. с англ. Конюшенко В.В. – Softline Co., 2014. – 74 с. Режим доступу: http://www.exponenta.ru/educat/free/matlab/gs.pdf .
11. Орлов А.И. Прикладная статистика. Учебник. / А.И. Орлов. – М.: Издательство “Экзамен”, 2004. – 656 с.

12. Растригин Л.А. Адаптация сложных систем. Методы и приложения / Л.А. Растригин. – Рига: Зинатне, 1981. – 375 с.

13. Ротштейн А.П. Интеллектуальные технологии идентификации / А.П. Ротштейн. – Винница: Вінниця–УНІВЕРСУМ, 1999. – 320 с.

14. Ротштейн О.П. Проектування нечітких баз знань: лабораторний практикум та курсове проектування: навч. посіб. / Ротштейн О.П., Штовба С.Д. – Вінниця: Вінницький державний технічний університет, 1999. – 65 с.

15. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с полськ / Рутковская Д., Пилинский М., Рутковский Л. – М.: Горячая линия – Телеком, 2006. – 452 с.

16. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика. / Уоссермен Ф. – М.: Мир, 1992. – 240 с.

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

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

19. Штовба С.Д. Генетичний алгоритм вибору правил нечіткої бази знань, збалансованої за критеріями точності та компактності / Штовба С.Д., Мазуренко В.В., Савчук Д.А. // Наукові праці Вінницького національного технічного університету. – 2012. – №3. – Режим доступу: http://praci.vntu.edu.ua/article/view/2335/2603 .

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

21. Штовба С.Д. Критерії точності та компактності для оцінювання якості нечітких баз знань в задачах ідентифікації / Штовба С.Д., Штовба О.В., Панкевич О.Д. // Наукові праці Вінницького національного технічного університету. – 2012. – №4. – Режим доступу: http://praci.vntu.edu.ua/article/view/2661/2872 .

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

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

24. Beale M.H. Neural Network Toolbox User's Guide / Mark Hudson Beale, Martin T. Hagan, Howard B. Demuth. – Mathworks Inc., 2014. – Режим доступу: http://www.mathworks.com/help/pdf_doc/nnet/nnet_ug.pdf

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

26. Leverington D. A Basic Introduction to Feedforward Backpropagation Neural Networks / David Leverington. – Texas Tech University, 2009. – Режим доступу: http://www.webpages.ttu.edu/dleverin/neural_network/neural_networks.html

27. Smith L. An Introduction to Neural Networks / Leslie Smith – University of Stirling, 2008. – http://www.cs.stir.ac.uk/~lss/NNIntro/InvSlides.html

28. 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
Мазуренко Віктор Володимирович

Інтелектуальні технології ідентифікації залежностей. Лабораторний практикум

Електронний навчальний посібник

Друкується в авторській редакції.

Оригінал-макет підготовлено авторами.

Формат 29,7х42¼.
Гарнітура Times New Roman
IEBN 804-479-000001-14

Вінницький національний технічний університет
навчально-методичний відділ ВНТУ
21021, м. Вінниця, Хмельницьке шосе, 95,
ВНТУ, к. 2201.
Тел. (0432) 59-87-36.
Свідоцтво суб'єкту 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 4 книг та біля 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

Мазуренко Віктор Володимирович

магістр, аспірант кафедри комп'ютерних систем управління Вінницького національного технічного університету. В 2011 р. закінчив Вінницький національний технічний університет за спеціальністю "Системи управління і автоматики".
Автор серії статей з проектування нечітких баз знань.

E-mail: viktor.mazurenko@gmail.com

IEBN 804-479-000001-14