6. Радіально базисні мережі

Радіально базисні нейронні мережі складаються з більшої кількості нейронів, ніж стандартні мережі з прямою передачею сигналів та навчанням методом зворотного поширення помилки, але на їх створення потрібно значно менше часу. Ці мережі особливо ефективні, коли є велика кількість навчальних векторів.

Нижче, крім мереж загального виду, обговорюються 2 спеціальні типи радіально базисних мереж: мережі GRNN (Generalized Regression Neural Networks) для вирішення завдань узагальненої регресії та мережі PNN (Probabilistic Neural Networks) для вирішення ймовірнісних завдань.

Для створення радіальних мереж загального вигляду призначені М-функції newrbe та newrb, а узагальнених регресійних та ймовірнісних – М-функції newgrnn та newpnn відповідно.

За командою help radbasis можна отримати наступну інформацію про М-функції, що входять до складу ППП Neural Network Toolbox і які стосуються побудови радіальних базисних мереж:

Radial basis networks	*Радіально базисні мережі*
New networks	**Формування мережі**
newrb	Створення радіально базисної мережі
newrbe	Створення радіально базисної мережі з нульовою помилкою
newgrnn	Створення узагальненої регресійної мережі
newpnn	Створення імовірнісної мережі
Using networks	Робота з мережею
sim	Моделювання мережі
Weight functions	Функції зважування
dist	Євклідова відстань
dotprod	Скалярний добуток
normprod	Нормований скалярний добуток
Net input functions	**Функції накопичення**
netprod	Добуток зважених входів
netsum	Сума зважених входів
Transfer функцій	**Функції активації**
compet	Конкуруюча функція активації
purelin	Функція активації з жорсткими обмеженнями
radbas	Радіально базисна функція активації
Performance	Функції оцінки якості мережі
mse	Середньоквадратична похибка
Signals	Перетворення даних
ind2vec	Перетворення індексного вектора на матрицю зв'язності
vec2ind	Перетворення матриці зв'язності на індексний вектор
Demonstrations	**Демонстраційні приклади**
demorb1	Радіально базисні мережі
demorb3	Приклад функцій активації, що не перекриваються
demorb4	Приклад функцій активації, що перекриваються
demogrn1	Мережа GRNN та апроксимація функцій
demopnn1	Мережа PNN та класифікація векторів

Модель нейрона та архітектура мережі

На рис. 1 показано радіально базисну мережу з R входами. Функція активації для радіально базисного нейрона має вигляд:

$$\text{radbas}(n) = e^{-n^2}.$$

Вхід функції активації визначається як модуль різниці вектора ваг *w* та вектора входу *p*, помножений на зміщення *b*.

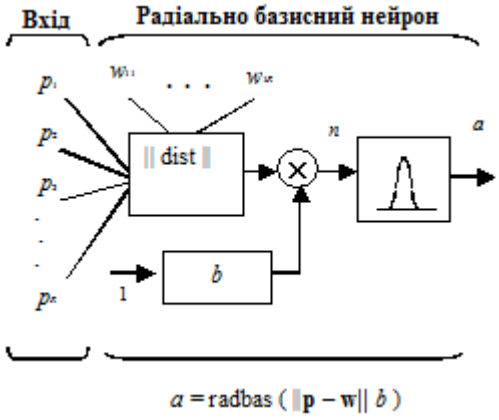


Рис. 1

Графік функції активації подано на рис. 2.

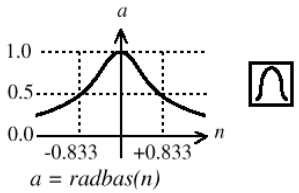


Рис. 2

Ця функція має максимум 1, коли вхід дорівнює 0. Коли відстань між векторами w і **p** зменшується, вихід радіально базисної функції збільшується. Таким чином, радіально базисний нейрон діє як індикатор, який формує значення 1 коли вхід p ідентичний вектору ваг w. Зміщення b дозволяє коригувати чутливість нейрона radbas. Наприклад, якщо нейрон має зміщення 0.1, його виходом буде 0.5 для будь-яких векторів входу p і ваги w таких, що відстань між ними дорівнює 8.333, або 0.833/b.

Радіально базисна мережа складається з двох прошарків: прихованого радіально базисного прошарку, що має S^1 нейронів, і вихідного лінійного прошарку, що має S^2 нейронів (рис. 3).

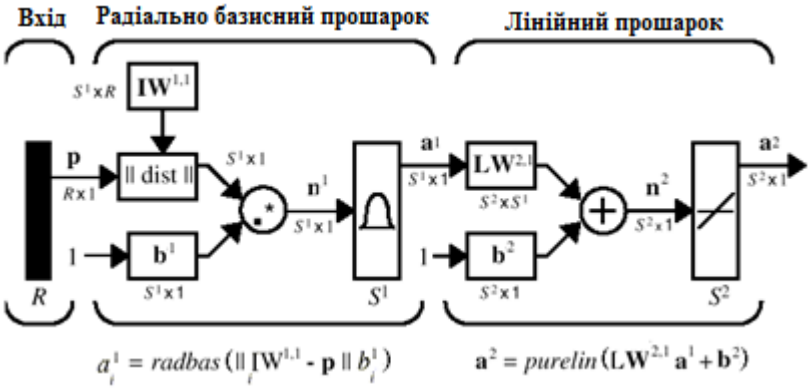


Рис. 3

Вхід блоку ||dist|| на цьому малюнку є вектор входу p і матриця ваг $IW^{1,1}$, а виходом - вектор, що складається з елементів S^1, які визначаються відстанями між i-м вектором входу p і i-й вектор-рядком $_iIW^{1,1}$ матриці ваг. Такий вектор-рядок називатимемо вектором ваг i-го нейрона. Вихід блоку | | dist | | множиться поелементно на вектор зміщення b^1 і формує вхід функції активації. Тоді вихід першого прошарку може бути записаний у такій формі:

```
a{1}=radbas(net.prod(dist(net.IW{1,1},p),net.b{1})))
```

Використовувати такий складний запис при застосуванні ППП Neural Network Toolbox не потрібно, оскільки всі операції, пов'язані зі створенням радіально базисної мережі, оформлені у вигляді спеціальних М-функцій newrbe та newrb.

Щоб зрозуміти поведінку мережі, необхідно простежити проходження вектора входу p. При заданні вектора входу кожен нейрон радіально базисного прошарку видасть значення відповідно до того, наскільки близький вектор входу до вектора ваг кожного нейрона. Таким чином, радіально базисні нейрони з векторами ваг, які значно відрізняються від вектора входу p, матимуть виходи, близькі до 0, і їх вплив на

виходи до лінійних нейронів буде незначним. Навпаки, радіально базисний нейрон з вектором ваг, близьким до вектора входу *p*, видасть значення, близьке до 1, і це значення буде передано на лінійний нейрон з вагою, що відповідає вихідному прошарку. Таким чином, якщо тільки 1 радіальний базисний нейрон має вихід 1, а всі інші мають виходи, рівні або дуже близькі до 0, вихід лінійного прошарку буде дорівнює вазі активного вихідного нейрона.

Створення мережі

Для побудови радіально базисних мереж призначені М-функції newrbe та newrb. Перша дозволяє побудувати радіально базисну мережу з нульовою помилкою, друга дозволяє керувати кількістю нейронів прихованого прошарку.

Радіально базисна мережа з нульовою помилкою

Для побудови радіально базисних мереж з нульовою помилкою призначена функція newrbe, яка викликається так:

```
net = newrbe(P,T,SPREAD)
```

Вхідними аргументами функції newrbe є масиви векторів входів *P* та цілей *T*, а також параметр впливу SPREAD. Вона повертає радіально базисну мережу з такими вагами і зсувами, що її виходи точно рівні цілям *T*. Функція newrbe створює стільки нейронів радіально базисного прошарку, скільки є вхідних векторів в масиві *P* і встановлює ваги першого прошарку рівними *P*'. При цьому зміщення встановлюються рівними 0.8326/SPREAD. Це означає, що рівень перекриття радіально базисних функцій дорівнює 0.5 та всі входи в діапазоні ±SPREAD є значними. Зрозуміло, що чим більший діапазон вхідних значень має бути прийнятий до уваги, тим більше значення параметра впливу SPREAD має бути встановлене. Це найбільш наочно проявляється під час вирішення завдань апроксимації функцій, у чому можна переконатися, звернувшись до демонстраційної програми demorb1.

Ваги другого прошарку IW^{21} (IW{2,1} – в позначеннях системи MATLAB) та зсувів b^2 (b{2}) можуть бути знайдені шляхом моделювання виходів першого прошарку a^1 (A{1}) та подальшого розв'язання системи лінійних алгебраїчних рівнянь (СЛАР)):

```
[W{2,1} b{2}] * [A{1}; ones] = T
```

Оскільки відомі входи другого прошарку A{1} і цілі T, а прошарок лінійний, то для обчислення ваги та зміщення другого прошарку достатньо скористатися розв'язувачем СЛАР

```
Wb = T/[P; ones(1, size(P,2))]
```

Тут матриця Wb містить ваги та зміщення (зміщення – в останньому стовпці). Сума квадратів похибок мережі завжди дорівнює 0, оскільки маємо завдання з Q рівняннями (пари вхід/ціль) і кожен нейрон має Q + 1 змінних (Q ваг за кількістю радіально базисних нейронів і одне зміщення). СЛАР з Q рівняннями та більш ніж Q змінними має вільні змінні та характеризується нескінченним числом розв'язків з нульовою похибкою.

Розглянемо приклад навчання та моделювання наступної радіально базисної мережі:

```
P = -1: 0.1 :1;
T = [-.9602 -.5770 -.0729 .3771 .6405 .6600 .4609 .1336 ...
-.2013 -.4344 -.5000 -.3930 -.1647 .0988 .3072 .3960 ...
.3449 .1816 -.0312 -.2189 -.3201];
plot(P,T,'*r','MarkerSize',4,'LineWidth',2)
hold on
% Створення мережі
net = newrbe(P,T); % Створення радіально базисної мережі
Warning: Rank deficient, rank = 13 tol = 2.2386e-014.
net.layers{1}.size % Число нейронів у прихованому прошарку
ans = 21
% Моделювання мережі
V = sim (net, P); % Вектори входу з навчальної множини
plot(P,V,'ob','MarkerSize',5, 'LineWidth',2)
p = [-0.75 -0.250.250.75];
v = sim (net, p); % Новий вектор входу
plot(p,v,'+k','MarkerSize',10, 'LineWidth',2)
```

Результати моделювання подано на рис. 4.

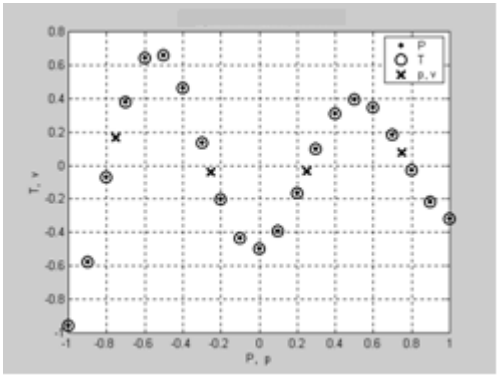


Рис. 4

Тут позначені значення входів *P*, цільових виходів *T*, і навіть результати обробки нового вектора *p*. Кількість використаних нейронів у прихованому прошарку в даному випадку дорівнює 21, що відповідає довжині навчальної множини.

Таким чином, в результаті навчання функція newrbe створює радіальну базисну мережу з нульовою похибкою на навчальній множині. Єдина умова, яку потрібно виконати, полягає в тому, щоб значення параметра SPREAD було достатньо великим, щоб активні області базисних функцій перекривалися, щоб покрити весь діапазон вхідних значень. Це дозволяє забезпечити необхідну гладкість апроксимуючих кривих та перешкоджає виникненню явища перенавчання. Однак значення SPREAD не повинен бути настільки великим, щоб радіально базисна функція оголошувала однаково значимими всі значення входу.

Недолік функції newrbe полягає в тому, що вона формує мережу з числом нейронів у прихованому прошарку, яке дорівнює кількості елементів навчальної множини. Тому за допомогою цієї функції не можна отримати прийнятного рішення у випадку великих розмірів навчальної множини, що характерно для реальних додатків.

Ітеративна процедура формування мережі

Функція newrb створює радіально базисну мережу, використовуючи ітеративну процедуру, яка додає по одному нейрону на кожному кроці. Нейрони додаються до прихованого прошарку до тих пір, поки сума квадратів помилок не стане меншою від заданого значення або не буде використано максимальну кількість нейронів. Ця функція викликається за допомогою команди

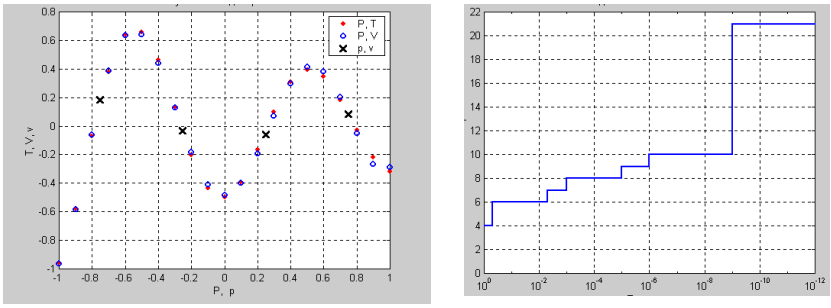
```
net = newrb(P,T,GOAL,SPREAD)
```

Входами функції newrb є масиви вхідних та цільових векторів *P* і *T*, а також параметри GOAL (допустима середньоквадратична помилка мережі), SPREAD (параметр впливу), а виходом – опис радіально базисної мережі. Значення параметра SPREAD має бути достатньо великим, щоб покрити весь діапазон значень входів, але не настільки, щоб ці значення були однаково значущими.

Застосуємо функцію newrb для створення радіально базисної мережі з попереднього прикладу.

```
P = -1:0.1:1;
T = [-.9602 -.5770 -.0729 .3771 .6405 .6600 .4609 .1336 ...
-.2013 -.4344 -.5000 -.3930 -.1647 .0988 .3072 .3960 ...
.3449 .1816 -.0312 -.2189 -.3201];
plot(P,T,'*r','MarkerSize',4,'LineWidth',2)
hold on
% Створення мережі
GOAL =0.01;% Допустиме значення функціоналу помилки
net = newrb(P,T,GOAL); % Створення радіально базисної мережі
net.layers{1}.size % Число нейронів у прихованому прошарку
ans = 6
% Моделювання мережі
V = sim (net, P); % Вектори входу з навчальної множини
plot(P,V,'ob','MarkerSize',5, 'LineWidth',2)
p = [-0.75 -0.25 0.25 0.75];
v = sim (net, p); % Новий вектор входу
plot(p,v,'+k','MarkerSize',10, 'LineWidth',2)
```

Відповідний графік подано на рис. 5 а.



а *б*

Рис. 6.5

Тут позначені значення входів *P*, цільових виходів *T*, і навіть результати обробки нового вектора *p*. Кількість нейронів у прихованому прошарку в даному випадку дорівнює шести, що відповідає значенню функціоналу помилки 0.01.На рис. 5 б показана залежність *кількості необхідних нейронів* прихованого прошарку від *точності* навчання. З цього графіка випливає, що для значень функціоналу помилки менше 1e-9 потрібно максимальну кількість нейронів, а це співпадає з числом нейронів радіально базисної мережі з нульовою похибкою.

Якщо порівнювати мережі з прямою передачею сигналу і радіально базисні мережі, слід зазначити, що у вирішенні одних і тих самих завдань вони мають певні переваги одна перед одною. Так, радіально базисні мережі з нульовою похибкою мають значно більше нейронів, ніж порівняна мережа з прямою передачею сигналу та сигмоїдальними функціями активації у прихованому прошарку. Це пов'язано з тим, що сигмоїдні функції активації перекривають більші діапазони значень входу, ніж радіально базисні функції. З іншого боку, проектування радіально базисної мережі вимагає значно меншого часу, а при обмеженій точності навчання може вимагати і меншої кількості нейронів, що використовуються.

Приклади радіально базисних мереж

Демонстраційний приклад demorb1 ілюструє застосування радіально базисної мережі для вирішення задачі апроксимації функції від однієї змінної.

Представимо функцію *f*(x) наступним розкладом:

$$f(x)=\sum_{i=1}^{N}\alpha_i\varphi_i(x), \qquad \textbf{(6.2)}$$

деφ_i(x) – радіально базисна функція.

Тоді ідея апроксимації може бути представлена графічно в такий спосіб. Розглянемо зважену суму трьох радіально базисних функцій, заданих інтервалі [–3 3].

```
p = -3:0.1:3;
a1 = radbas(p);
a2 = radbas(p-1.5);
a3 = radbas(p+2);
a = a1 + a2 * 1 + a3 * 0.5;
plot (p, a1, p, a2, p, a3 * 0.5, p, a) % Рис. 6
```

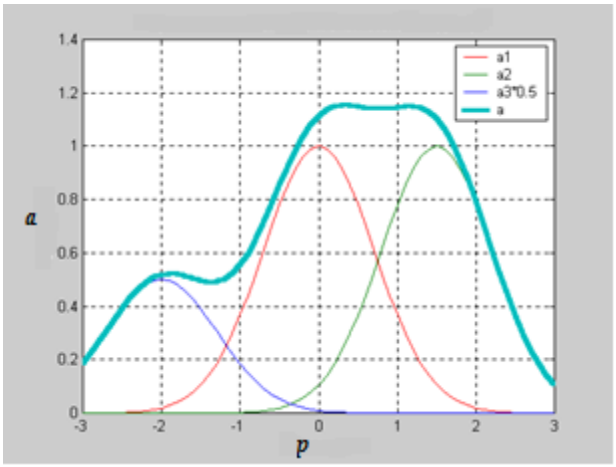


Рис. 6

Як випливає з аналізу рис. 6 розкладання за радіально базисними функціями забезпечує необхідну гладкість. Тому їх застосування для апроксимації довільних нелінійних залежностей цілком виправдане. Такий розклад може бути реалізовано за допомогою двошарової нейронної мережі, перший прошарок якої

складається з радіально базисних нейронів, а другий – з єдиного нейрона з лінійною характеристикою, на якому реалізовується додавання виходів нейронів першого прошарку.

Приступимо до формування радіально базисної мережі. Сформуємо навчальну множину і задамо допустиме значення функціоналу помилки, що дорівнює 0.01,параметр впливу визначимо рівним 1 і використовуватимемо ітераційну процедуру формування радіально базисної мережі:

```
P = -1: 0.1:1;
T = [-.9602 -.5770 -.0729 .3771 .6405 .6600 .4609 .1336 ...
-.2013 -.4344 -.5000 -.3930 -.1647 .0988 .3072 .3960 ...
.3449 .1816 -.0312 -.2189 -.3201];
GOAL =0.01;% Допустиме значення функціоналу помилки
SPREAD = 1; % Параметр впливу
net = newrb(P,T,GOAL,SPREAD); % Створення мережі
net.layers{1}.size % Число нейронів у прихованому прошарку
ans = 6
```

Для заданих параметрів нейронна мережа складається із шести нейронів і після навчання забезпечує можливость апроксимації нелінійних залежностей. Моделюючи сформовану нейронну мережу, побудуємо апроксимаційну криву на інтервалі [–1 1] з кроком 0.01 для нелінійної залежності.

```
plot(P,T,'+k') % Точки навчальної множини
hold on;
X = -1: .01: 1;
Y = sim (net, X); % Моделювання мережі
plot(X,Y); % Рис. 7
```

З аналізу рис. 7 слідує, що при невеликій кількості нейронів прихованого прошарку радіально базисна мережа досить добре апроксимує нелінійну залежність, задану навчальною множиною з 21 точки.

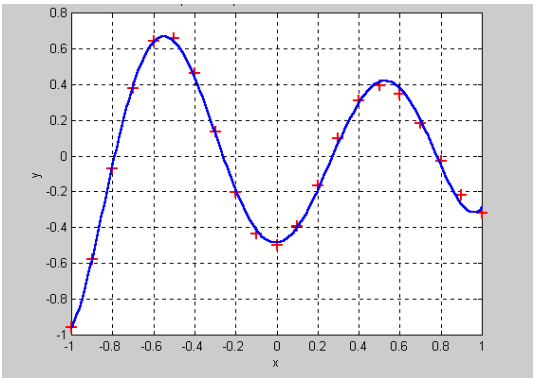


Рис. 7

У демонстраційних прикладах demorb3 та demorb4 досліджується вплив параметра SPREAD на структуру радіально базисної мережі та якість апроксимації. У демонстраційному прикладі demorb3 параметр впливу SPREAD встановлено рівним 0.01.Це означає, що діапазон перекриття вхідних значень становить лише ±0.01, а оскільки навчальні входи задані з інтервалом 0.1, вхідні значення функціями активації не перекриваються.

```
GOAL =0.01;% Допустиме значення функціоналу помилки
SPREAD =0.01;% Параметр впливу
net = newrb(P,T,GOAL,SPREAD); % Створення мережі
net.layers{1}.size % Число нейронів у прихованому прошарку
ans = 19
```

Це приводить до того, що, по-перше, збільшується кількість нейронів прихованого прошарку з 6 до 19, а по-друге, не забезпечується необхідної гладкості функції, що апроксимується:

```
plot(P,T,'+k') % Точки навчальної множини
hold on;
X = -1: .01: 1;
Y = sim (net, X); % Моделювання мережі
plot(X,Y); % Рис. 8
```

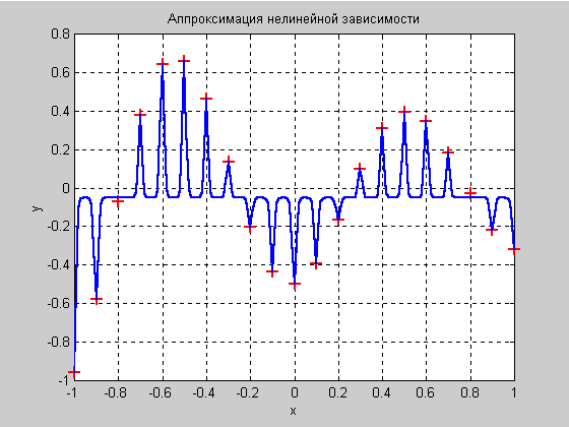


Рис. 8

Приклад demorb4 ілюструє протилежний випадок, коли параметр впливу SPREAD вибирається досить великим (в даному прикладі – 12 або більше), всі функції активації перекриваються і кожен базисний нейрон видає значення, близьке до 1, для всіх значень входів. Це приводить до того, що мережа не реагує на вхідні значення. Функція newrb намагатиметься будувати мережу, але не зможе забезпечити необхідної точності через обчислювальні проблеми.

```
GOAL =0.01;% Допустиме значення функціоналу помилки
SPREAD = 12; % Параметр впливу
net = newrb(P,T,GOAL,SPREAD); % Створення мережі
net.layers{1}.size % Число нейронів у прихованому прошарку
ans = 21
```

У процесі обчислень виникають труднощі з обертанням матриць, і щодо цього видаються попередження; кількість нейронів прихованого прошарку встановлюється рівною 21, а точність апроксимації виявляється неприпустимо низькою:

```
plot(P,T,'+k') % Точки навчальної множини
hold on;
X = -1: 0.01: 1;
Y = sim (net, X); % Моделювання мережі
plot(X,Y); % Рис. 9
```

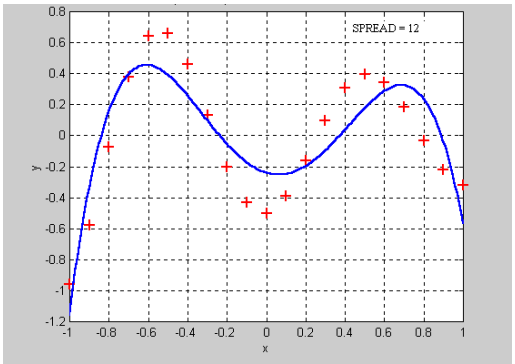


Рис. 9

Висновок з виконаного дослідження полягає в тому, що параметр впливу SPREAD слід вибирати більшим, ніж крок розбиття інтервалу задання навчальної послідовності, але меншим за розмір інтервалу. Для цієї задачі це означає, що параметр впливу SPREAD повинен бути більшим за 0.1 і меншим за 2.

6.1. Мережі GRNN

Нейронні мережі GRNN (Generalized Regression Neural Network) призначені для вирішення завдань узагальненої регресії, аналізу часових рядів та апроксимації функцій. Характерною особливістю цих мереж є дуже висока швидкість навчання.

Архітектура мережі

Архітектура мережі GRNN показана на рис. 10. Вона аналогічна архітектурі радіально базисної мережі, але відрізняється структурою другого прошарку, в якому використовується блок normprod для обчислення нормованого скалярного добутку рядка масиву ваг LW^{21} та вектора входу a^1 відповідно до наступного співвідношення:

$$\mathbf{n}^2 = \frac{\mathbf{LW}^{21}\mathbf{a}^1}{sum(\mathbf{a}^1)}.$$

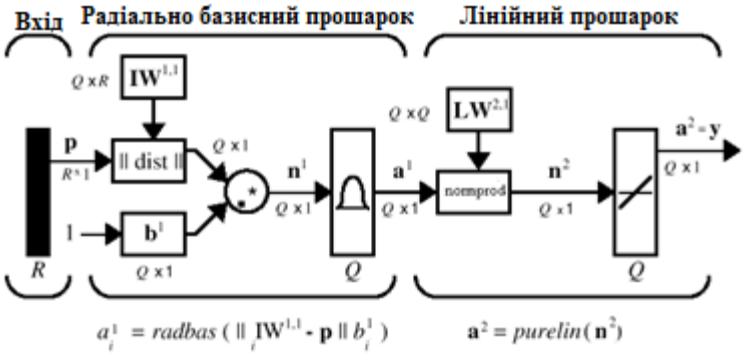


Рис. 10

Перший прошарок - це радіально базисний прошарок з числом нейронів, рівним числу Q елементів навчальної множини; як початкове наближення для матриці ваг вибирається масив *P*'; зміщення b^1 встановлюється рівним вектор-стовпцю з елементами 0.8326/SPREAD. Функція **dist** обчислює відстань між вектором входу та вектором ваги нейрона; вхід функції активації n^1 дорівнює поелементному добутку зваженого входу мережі на вектор зміщення; вихід кожного нейрона першого прошарку a^1 є результатом перетворення вектора n^1 радіальною базисною функцією **radbas**. Якщо вектор ваги нейрона дорівнює транспонованому вектору входу, то зважений вхід дорівнює 0, а вихід функції активації - 1. Якщо відстань між вектором входу і вектором ваги нейрона дорівнює spread, вихід функції активації буде дорівнює 0.5.

Другий прошарок - це лінійний прошарок з числом нейронів, також рівним числу елементів Q навчальної множини, причому як початкове наближення для матриці ваг LW{2,1} вибирається масив *T*. Припустимо, що маємо вектор входу p_i, близький до одного з векторів входу *p* з навчальної множини. Цей вхід p_i генерує значення виходу прошарку a_i^1, близьке до 1. Це приводить до того, що вихід прошарку 2 буде близький до t_i.

Якщо параметр впливу SPREAD малий, то радіально базисна функція характеризується різким спадом і діапазон вхідних значень, на який реагують нейрони прихованого прошарку, виявляється дуже малим. Зі збільшенням параметра SPREAD нахил радіально базисної функції стає гладкішим, і в цьому випадку вже кілька нейронів реагують на значення вектора входу. Тоді на виході мережі формується вектор, що відповідає середньому арифметичному кількох цільових векторів, відповідних вхідним векторам навчальної множини, близьким до вектора входу. Чим більше значення параметра SPREAD, тим більше нейронів бере участь у формуванні середнього значення, і в результаті функція, що генерується мережею, стає більш гладкою.

Синтез мережі

Для створення нейронної мережі GRNN призначено М-функцію newgrnn. Задамо таку навчальну множину векторів входу та цілей і побудуємо мережу GRNN:

```
P = [4 5 6];
T = [1.5 3.6 6.7];
net = newgrnn(P,T);
net.layers{1}.size % Число нейронів у прихованому прошарку
ans = 3
```

Ця мережа має 3 нейрони в прихованому прошарку. Промоделюємо побудовану мережу спочатку для одного входу, а потім для послідовності входів з інтервалу [4 7]:

```
p = 4.5;
v = sim (net, p);
p1 = 4: 0.1: 7;
v1 = sim (net, p1);
plot(P,T,'*k',p,v,'ok',p1,v1,'-k','MarkerSize',10,'LineWidth',2)
```

Результат показано на рис. 11.

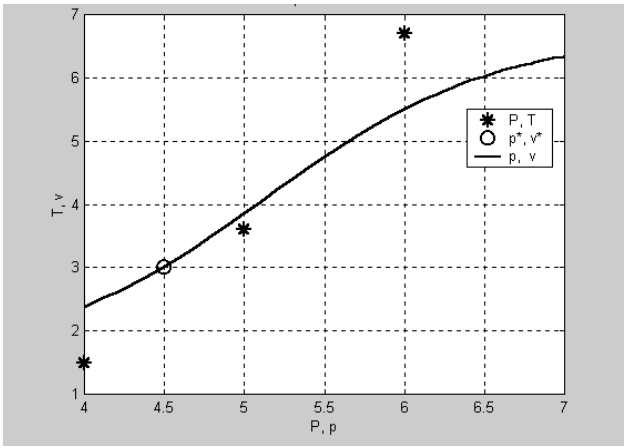


Рис. 11

Зауважимо, що для мережі GRNN розмір вектора, що вводиться, може відрізнятися від розміру векторів, що використовуються в навчальній послідовності. Крім того, в даному випадку апроксимуюча функція може значно відрізнятися від значень, що відповідають навчальній послідовності.

Демонстраційна програма demogrn1 ілюструє, як мережі GRNN вирішують завдання апроксимації. Визначимо навчальну множину у вигляді масивів *P* і *T*.

```
P = [1 2 3 4 5 6 7 8];
T = [0 1 2 3 2 1 2 1];
```

Для створення мережі GRNN використовується функція newgrnn. Візьмемо значення параметра впливу SPREAD трохи меншим, ніж крок задання аргументу функції (в даному випадку 1), щоб побудувати апроксимуючу криву, близьку до заданих точок. Як ми вже бачили раніше, чим менше значення параметра SPREAD, тим ближче точки апроксимуючої кривої до заданих, але тим менш гладкою є сама крива:

```
spread = 0.7;
net = newgrnn (P, T, spread);
net.layers{1}.size % Число нейронів у прихованому прошарку
ans = 8
A = sim (net, P);
plot(P,T,'*k','markersize',10)
hold on,
plot(P,A,'ok','markersize',10);
```

Результат показано на рис. 12.

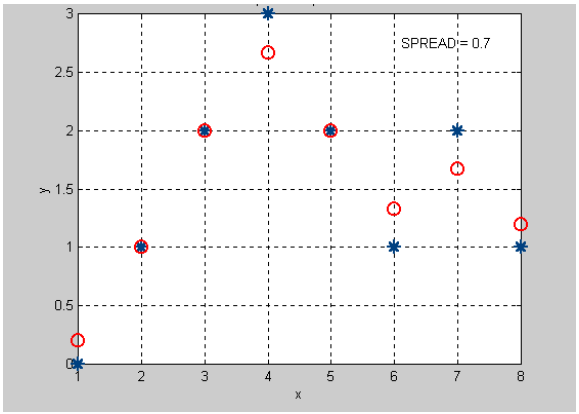


Рис.12

Моделювання мережі для діапазону значень аргументу дозволяє побачити всю апроксимуючу криву, причому можлива екстраполяція цієї кривої за межі області її визначення. Для цього задамо інтервал аргументу в діапазоні [–1 10]:

```
P2 = -1: 0.1: 10;
A2 = sim (net, P2);
plot(P2,A2,'-k','linewidth',2)
hold on,
plot(P,T,'*k','markersize',10)
```

Результат показано на рис. 13.

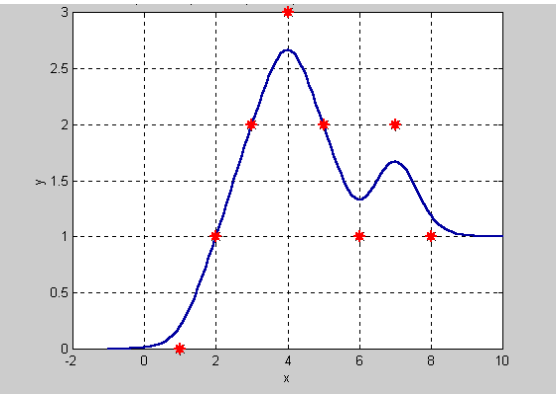


Рис. 13

Сформована мережа GRNN використовує всього 8 нейронів у прихованому прошарку і дуже успішно вирішує задачу апроксимації та екстраполяції нелінійної залежності, що відновлюється за експериментальними точками.

6.2. Мережі PNN

Нейронні мережі PNN (Probabilistic Neural Networks) призначені для вирішення імовірнісних завдань, зокрема задач класифікації.

Архітектура мережі

Архітектура мережі PNN базується на архітектурі радіально базисної мережі, але як другий прошарок використовує так званий конкуруючий прошарок, який підраховує ймовірність приналежності вхідного вектора до того чи іншого класу і в кінцевому рахунку зіставляє вектор з тим класом, ймовірність приналежності до якого є більшою. Структура мережі PNN подана рис. 14.

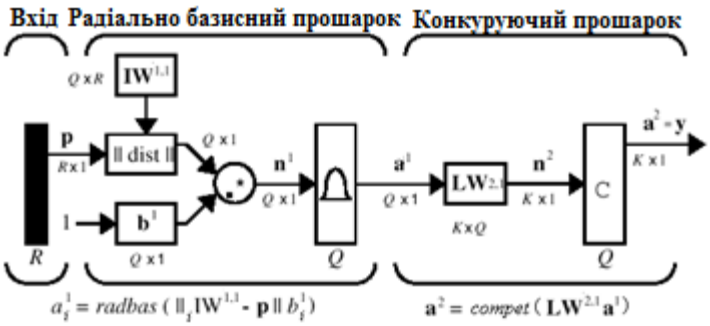


Рис. 14

Передбачається, що задано навчальну множину, що складається з Q пар векторів вхід/ціль. Кожен цільовий вектор має *K* елементів, що вказують на клас приналежності, і, таким чином, кожен вектор входу належатиме до одного з *K* класів. В результаті може бути утворена матриця зв'язності *T* розміру *K*×*Q*, що складається з нулів та одиниць, рядки якої відповідають класам належності, а стовпці – векторам входу. Таким чином, якщо елемент T(i, j) матриці зв'язності дорівнює 1, це означає, що *j*-й вхідний вектор належить до *i*-го класу.

Вагова матриця першого прошарку IW^{11} (net.IW{1,1}) формується з використанням векторів входу з навчальної множини у вигляді матриці *P*'. Коли подається новий вхід, блок ||dist|| обчислює близькість нового вектора до векторів навчальної множини; Потім обчислені відстані множаться на зміщення та подаються на вхід функції активації radbas. Вектор навчальної множини, найбільш близький до вектора входу, буде представлений у векторі виходу a^1 числом, близьким до 1.

Вагова матриця другого прошарку LW^{21} (net.LW{2,1}) відповідає матриці зв'язності *T*, побудованій для даної навчальної послідовності. Ця операція може бути виконана за допомогою M-функції ind2vec, яка перетворює цільовий вектор на матрицю зв'язності *T*. Добуток $T*a^1$ визначає елементи вектора a^1, що відповідають кожному з *K* класів. В результаті конкуруюча функція активації другого прошарку compet формує на виході значення, що дорівнює 1, для найбільшого за величиною елемента вектора n^2 і 0 в інших випадках. Таким чином, мережа PNN виконує класифікацію векторів входу на *K* класів.

Синтез мережі

Для створення нейронної мережі PNN призначено M-функцію newpnn. Визначимо 7 наступних векторів входу та віднесемо кожен з них до одного із трьох класів:

```
P = [0 0;1 1;0 3;1 4;3 1;4 1;4 3]';
Tc = [1 1 2 2 3 3 3];
```

Вектор T_c назвемо вектором індексів класів. Цьому індексному вектору можна поставити у відповідність матрицю зв'язності T у вигляді розрідженої матриці виду

```
T = ind2vec(Tc)
T =
(1,1)    1
(1,2)    1
(2,3)    2
(2,4)    2
(3,5)    3
(3,6)    3
(3,7)    3
```

яка визначає належність перших двох векторів до класу 1, двох наступних –до класу 2 та трьох останніх – до класу 3. Повна матриця T має вигляд:

```
T = full (T)
T =
1 1 0 0 0 0 0
0 0 1 1 0 0 0
0 0 0 0 1 1 1
```

Масиви P і T задають навчальну множину, що дозволяє виконати формування мережі, промоделювати її, використовуючи масив входів P, і переконатися, що мережа правильно вирішує задачу класифікації на елементах навчальної множини. В результаті моделювання мережі формується матриця зв'язності, що відповідає масиву векторів входу. Для того щоб перетворити її на індексний вектор, призначена M-функція vec2ind:

```
net = newpnn(P,T);
net.layers{1}.size % Число нейронів у мережі PNN
ans = 7
Y = sim (net, P);
Yc = vec2ind(Y)
Yc = 1 1 2 2 3 3 3
```

Результат підтверджує правильність розв'язання задачі класифікації.

Виконаємо класифікацію деякого набору довільних векторів p, що не належать навчальній множині, використовуючи раніше створену мережу PNN:

```
p = [13; 0 1; 5 2]';
```

Виконуючи моделювання мережі для цього набору векторів, отримуємо

```
a = sim (net, p);
ac = vec2ind(a)
ac = 2 1 3
```

Фрагмент демонстраційної програми demopnn1 дозволяє проілюструвати результати класифікації у графічному вигляді:

```
clf reset, drawnow
p1 = 0:0.05:5;
p2 = p1;
[P1, P2] = meshgrid (p1, p2);
pp = [P1(:) P2(:)];
aa = sim (net, pp');
aa = full (aa);
m = mesh(P1,P2,reshape(aa(1,:),length(p1),length(p2))));
set(m,'facecolor',[0.75 0.75 0.75],'linestyle','none');
hold on
view(3)
m = mesh(P1,P2,reshape(aa(2,:),length(p1),length(p2))));
set(m,'facecolor',[0 1 0.5],'linestyle','none');,
m = mesh(P1,P2,reshape(aa(3,:),length(p1),length(p2)));
set(m,'facecolor',[0 1 1],'linestyle','none');
plot3(P(1,:),P(2,:),ones(size(P,2))+0.1,'.','markersize',30)
plot3(p(1,:),p(2,:),1.1*ones(size(p,2)),'*','markersize',20,...'color',[1 0 0])
hold off
view(2)
```

Результати класифікації, які проілюстровано на рис. 15, показують, що 3 подані мережі вектори (відмічені зірочками) класифікуються мережею PNN (що складається з семи нейронів) абсолютно правильно.

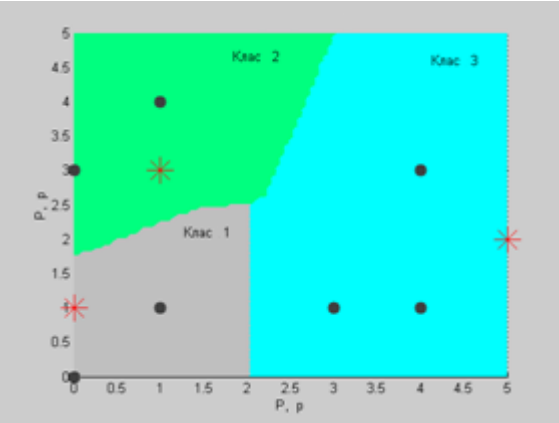


Рис. 15

Насамкінець зазначимо, що мережі PNN можуть дуже ефективно застосовуватися для вирішення завдань класифікації. Якщо задано досить велику навчальну множину, то розв'язок, що генеруються мережами, збігається до розв'язків, що відповідають правилу Байєса. Недолік мереж GRNN і PNN полягає в тому, що вони працюють відносно повільно, оскільки виконують дуже великі обсяги обчислень у порівнянні з іншими типами нейронних мереж.