Рекурентні мережі

У цьому розділі розглядаються два типи рекурентних нейронних мереж, що становлять найбільший інтерес для користувачів, - це клас мереж Елмана (Elman) і клас мереж Хопфілда (Hopfield). Характерною особливістю архітектури рекурентної мережі є наявність блоків динамічної затримки та зворотних зв'язків. Це дозволяє таким мережам обробляти динамічні моделі. Звернемося до опису конкретних типів рекурентних нейронних мереж.

Мережі Елмана

Мережа Елмана - це мережа, що складається з двох прошарків, в якій прихований шар охоплений динамічним зворотним зв'язком. Це дозволяє врахувати передісторію процесів і накопичити інформацію для вироблення правильної стратегії управління. Мережі Елмана застосовуються в системах управління об'єктами, що рухаються, при побудові систем технічного зору та в інших додатках.

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

Elman recurrent networks	*Рекурентні мережі Елмана*
New networks	**Формування мережі**
newelm	Створення мережі Елмана
Using networks	**Робота з мережею**
sim init adapt train	Моделювання Ініціалізація Адаптація Навчання
Weight functions	**Функції зважування**
dotprod ddotprod	Скалярний добуток Похідна скалярного добутку
Net input functions	**Функції накопичення**
netsum dnetsum	Сума зважених входів Похідна суми завислих входів
Transfer функцій	**Функції активації**
purelin tansig logsig dpurelin dtansig dlogsig	Лінійна Гіперболічний тангенс Логістична Похідна лінійної функції Похідна гіперболічного тангенсу Похідна логістичної функції
Performance functions	**Функції оцінки якості мережі**
mse msereg dmse dmsereg	Середньоквадратична помилка навчання Середньоквадратична помилка навчання при застосуванні регуляризації Похідна середньоквадратичної помилки навчання Похідна середньоквадратичної помилки навчання при застосуванні регуляризації
Initialization функцій	**Функції ініціалізації мережі**
initlay	Пошарова ініціалізація

initnw	Функція NW (Nguyen – Widrow)
Learning functions	**Опції настроювання параметрів**
learngd learngdm	Функція налаштування методом градієнтного спуску Функція налаштування методом градієнтного спуску зі збуренням
Adapt functions	**Функції адаптації**
adapt	Адаптація ваг та зміщень
Training functions	**Функції навчання**
traingd traingdm traingda	Градієнтний спуск за правилом зворотного розповсюдження помилки Градієнтний спуск зі збуренням Градієнтний спуск із адаптацією параметра швидкості налаштування
Demonstrations	**Демонстраційні приклади**
appelm1	Приклад рекурентної мережі Елмана

Архітектура

Мережа Елмана - це, як правило, двошарова мережа із зворотним зв'язком від виходу до входу першого прошарку (рис. 1).

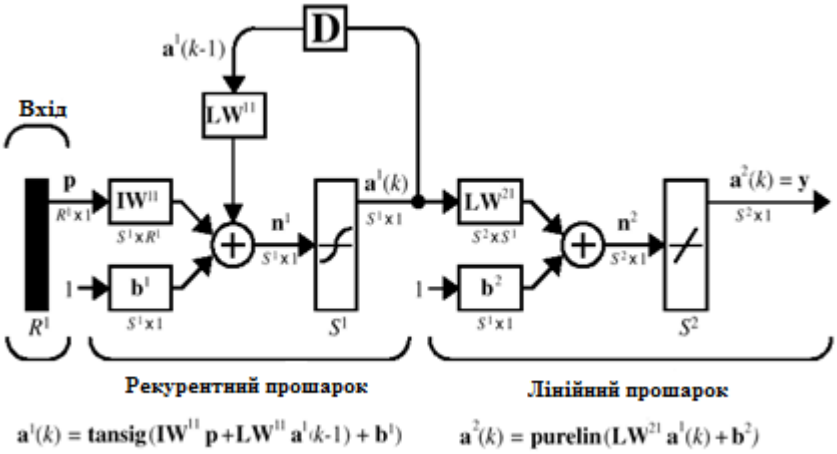


Рис. 1

Як функції активації в мережі Елмана часто використовуються: у прихованому, рекурентному прошарку – функція гіперболічного тангенсу tansig, у лінійному прошарку – функція purelin. Таке поєднання функцій активації дозволяє максимально точно апроксимувати функції скінченного числа точок розриву. Єдина вимога до мережі, полягає в тому, щоб прихований прошарок мав досить велику кількість нейронів, що необхідно для успішної апроксимації складних функцій.

Відповідно до структурної схеми мережі Елмана сформуємо динамічний опис її рекурентного прошарку у вигляді рівнянь стану

$$\begin{cases} \mathbf{n}^1(k) = \mathbf{LW}^{11}\mathbf{a}^1(k-1) + \mathbf{IW}^{11}\mathbf{p} + \mathbf{b}^1, \quad \mathbf{a}^1(0) = \mathbf{a}_0^1; \\ \mathbf{a}^1(k) = \text{tansig}(\mathbf{n}^1(k)). \end{cases}$$

Ця рекурентна матрична форма рівнянь стану вкотре підкреслює назву нейронних мереж, що вивчаються.

Другий, лінійний прошарок є безінерційним та описується співвідношеннями

$$\begin{cases} n^2(k) = \mathbf{LW}^{21}\mathbf{a}^1(k) + \mathbf{b}^2; \\ \mathbf{a}^2(k) = \text{purelin}(\mathbf{n}^2(k)). \end{cases}$$

Нижче мережа Елмана досліджується на прикладі завдання виявлення амплітуди гармонійного сигналу. Нехай відомо, що на вхід нейронної мережі надходять вибірки з деякого набору синусоїд. Потрібно виділити значення амплітуд цих синусоїд.

Далі розглядаються вибірки з набору двох синусоїд з амплітудами 1.0 та 2.0:

```
p1 = sin(1:20);
p2 = sin(1:20)*2;
```

Цільовими виходами такої мережі є вектори

```
t1 = ones (1,20);
t2 = ones (1,20) * 2;
```

Сформуємо набір векторів входу та цільових виходів:

```
p = [p1 p2 p1 p2];
t = [t1 t2 t1 t2];
```

Сформуємо навчальну послідовність у вигляді масивів комірок:

```
Pseq = con2seq (p);
Tseq = con2seq(t);
```

Створення мережі

У ППП NNT для створення мережі Елмана передбачена M-функція newelm. Завдання вище вимагає, щоб мережа Елмана на кожному кроці спостереження значень вибірки могла виявити єдиний її параметр - амплітуду синусоїди. Це означає, що мережа повинна мати 1 вхід та 1 вихід:

```
R = 1; % Число елементів входу
S2 = 1; % Число нейронів вихідного прошарку
```

Рекурентний прошарок може мати будь-яке число нейронів, і чим складніше завдання, тим більше нейронів потрібно. Зупинимося на 10 нейронах рекурентного прошарку:

```
S1 = 10; % Число нейронів рекурентного прошарку
```

Елементи входу для цієї задачі змінюються в діапазоні від –2 до 2. Для навчання використовується метод градієнтного спуску зі збуренням та адаптацією параметра швидкості налаштування, реалізований у вигляді M-функції traingdx:

```
net = newelm([-2 2],[S1 S2],{'tansig','purelin'},'traingdx');
```

Мережа використовує такі функції адаптації, ініціалізації, навчання та оцінки якості:

```
adaptFcn: 'adaptwb'
initFcn: 'initlay'
performFcn: 'mse'
trainFcn: 'traingdx'
```

Прошарки мережі Елмана мають такі характеристики:

```
net.layers{1}
ans =
dimensions: 10
distanceFcn: 'dist'
distances: [10×10 double]
initFcn: 'initnw'
netInputFcn: 'netsum'
positions: [0 1 2 3 4 5 6 7
8 9]
size: 10
```

```
net.layers{2}
ans =
dimensions: 1
distanceFcn: 'dist'
distances: 0
initFcn: 'initnw'
netInputFcn:
'netsum'
positions: 0
size: 1
```

```
topologyFcn: 'hextop'          topologyFcn:
transferFcn: 'tansig'         'hextop'
userdata: [1×1 struct]         transferFcn:
                              'purelin'
                               userdata:       [1×1
                              struct]
```

Прихований прошарок використовує функцію активації tansig, яка для мережі Елмана приймається за замовчуванням; ініціалізація ваг та зміщень реалізовується методом NW (Nguen – Widrow) за допомогою М-функції initnw. Другий прошарок використовує лінійну функцію активації purelin.

За замовчуванням для налаштування ваг та зсувів використовується функція learngdm, а для оцінки якості навчання – функція mse.

Навчання мережі

Для навчання мережі Елмана можуть бути використані як процедура адаптації, так і процедура навчання, що реалізовуються за допомогою функцій adapt і train відповідно.

У процесі процедури адаптації на кожному кроці виконуються такі дії:

- моделювання мережі при подачі повного набору векторів входу та обчислення помилки мережі;
- обчислення наближеного градієнта функції помилки, залежної від ваг та зміщень, методом зворотного поширення помилки;
- налаштування ваг з використанням функції налаштування, що вибирається користувачем; рекомендується функція learngdm.

У процесі процедури навчання на кожному циклі виконуються такі дії:

- моделювання мережі при подачі послідовності вхідних сигналів, порівняння з цільовими виходами та обчислення помилки;
- обчислення наближеного градієнта функції помилки, залежної від ваг та зміщень, методом зворотного поширення помилки;
- налаштування ваг з використанням функції налаштування, що вибирається користувачем; Рекомендується функція traingdx.

Мережі Елмана не забезпечують високої точності розв'язку, оскільки присутність зворотного зв'язку в рекурентному прошарку не дозволяє точно обчислити градієнт функціоналу.

Надалі для навчання мережі Елмана використовується М-функція train. Її вхідними аргументами є навчальні послідовності Pseq і Tseq, як метод навчання використовується метод зворотного поширення помилки зі збуренням та адаптацією параметра швидкості налаштування. Кількість циклів навчання приймається рівною 1000, періодичність виведення результатів – 20 циклів, кінцева похибка навчання –0.01:

```
net.trainParam.epochs = 1000;
net.trainParam.show =25;
net.trainParam.goal =0.01;
[net, tr] = train (net, Pseq, Tseq);
```

Після 500 циклів навчання отримаємо наступний графік помилки (рис. 2).

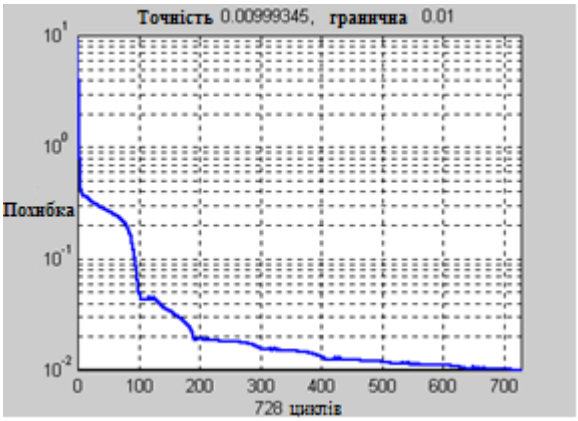


Рис. 2

Необхідна точність навчання забезпечується за 728 циклів. Тепер можна перевірити роботу сформованої мережі.

Перевірка мережі

Будемо використовувати для перевірки мережі входи навчальної послідовності:

```
figure(2)
a = sim (net, Pseq);
time = 1: length (p);
plot(time, t,'--', time, cat(2,a{:}))
axis([1 80 0.8 2.2]) % Рис.3
```

На рис. 3 наведено графіки вхідного та вихідного сигналів.

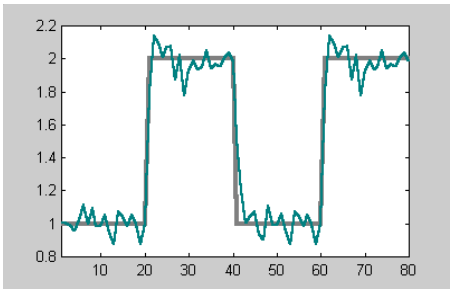


Рис. 3

Як випливає з аналізу малюнка, мережа справляється з розв'язанням задачі виявлення амплітуди на наборах навчальної множини. Однак незрозуміло, як вона поводитиметься на інших наборах входу. Чи має побудована мережа Елмана властивість узагальнення? Спробуємо перевірити це, виконавши такі дослідження.

Подамо на мережу набір сигналів, складений з двох синусоїд з амплітудами 1.6 та 1.2 відповідно:

```
p3 = sin(1:20)*1.6;
t3 = ones (1,20) * 1.6;
p4 = sin(1:20)*1.2;
t4 = ones (1,20) * 1.2;
pg = [p3 p4 p3 p4];
tg = [t3 t4 t3 t4];
pgseq = con2seq (pg);
figure(3)
a = sim (net, pgseq);
time = 1: length (pg);
```

```
plot(time, tg,'--', time, cat(2,a{:}))
axis([1 80 0.8 2.2])
```

Результат подано на рис. 4.

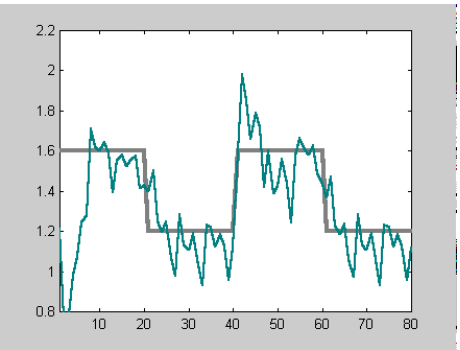


Рис. 4

Цього разу мережа гірше справляється із завданням. Мережа прагне детектувати значення амплітуди, але робить це дуже точно. Поліпшене узагальнення могло б бути отримане при навчанні мережі більшої кількості амплітуд, ніж тільки значення 1.0 і 2.0. Використання трьох або чотирьох гармонійних сигналів з різними амплітудами може привести до кращого датчика амплітуд.

Читач може продовжити вивчення мереж Елмана, використовуючи програму appelm1. Зробивши копію цієї програми, можна продовжити експерименти, збільшуючи кількість нейронів у рекурентному прошарку або тривалість, а також кількість вхідних наборів.

Мережі Хопфілда

Будь-який цільовий вектор можна розглядати як набір характерних ознак деякого об'єкта. Якщо створити рекурентну мережу, положення рівноваги якої збігалося б із цим цільовим вектором, то таку мережу можна було б розглядати як асоціативну пам'ять. Надходження на вхід такої мережі деякого набору ознак у вигляді початкових умов приводило б її до того чи іншого положення рівноваги, що дозволяло б асоціювати вхід з деяким об'єктом. Саме такі асоціативні можливості й мають мережі Хопфілда. Вони відносяться до класу рекурентних нейронних мереж, що володіють тією властивістю, що за скінченне число тактів часу вони з довільного початкового стану приходять у стан стійкої рівноваги, що називається атрактором. Кількість таких атракторів визначає об'єм асоціативної пам'яті мережі Хопфілда.

Спроектувати мережу Хопфілда - це означає створити рекурентну мережу з певної множиною точок рівноваги, таких, що при заданні початкових умов мережа в кінцевому рахунку приходить у стан спокою в одній із цих точок. Властивість рекурсії проявляється у тому, що вихід мережі подається назад на вхід. Можна сподіватися, що вихід мережі встановиться в одній із точок рівноваги. Пропонований нижче метод синтезу мережі Хопфілда не є абсолютно досконалим у тому сенсі, що мережа, що синтезується, на додаток до бажаних може мати паразитні точки рівноваги. Однак кількість таких паразитних точок має бути зведена до мінімуму за рахунок конструювання методу синтезу. Більше того, область притягування точок рівноваги має бути максимально великою.

Метод синтезу мережі Хопфілда заснований на побудові системи лінійних диференціальних рівнянь першого порядку, яка задана в деякому замкнутому гіперкубі простору станів і має розв'язок у вершинах цього гіперкуба. Така мережа дещо відрізняється від класичної моделі Хопфілда, але вона простіша для розуміння та проектування, і ми посилатимемося на неї як на модифіковану мережу Хопфілда.

За командою help hopfield можна отримати наступну інформацію про M-функції, що входять до складу ППП Neural Network Toolbox і які стосуються побудови модифікованих мереж Хопфілда:

Hopfield recurrent networks	*Рекурентна модифікована мережа Хопфілда*
New networks	**Формування мережі**
newhop	Створення модифікованої мережі Хопфілда
Weight functions	**Операції з ваговою функцією**
dotprod	Скалярний добуток
Net input functions	**Операції над входами**
netsum	Підсумовування
Transfer функцій	**Функції активації**
satlins	Симетрична лінійна функція з обмеженнями
Demonstrations	**Демонстраційні приклади**
demohop1	Приклад двовимірної модифікованої мережі Хопфілда
demohop2	Приклад нестійкої точки рівноваги
demohop3	Приклад тривимірної модифікованої мережі Хопфілда
demohop4	Приклад стійких паразитних точок рівноваги

Архітектура мережі

Архітектура модифікованої мережі Хопфілда представлена на рис. 5.

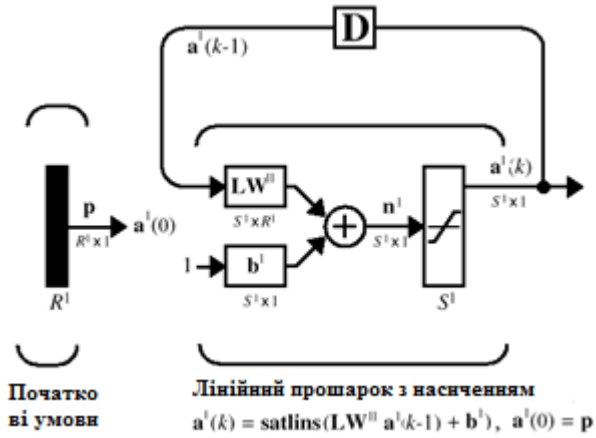


Рис. 5

Вхід *p* установлює значення початкових умов. У мережі використовується лінійна функція активації з насиченням satlins, яка описується наступним чином:

$$a = satlins(n) = \begin{cases} -1, & n < -1; \\ n, & -1 \le n \le 1; \\ 1, & n > 1. \end{cases}$$

Ця мережа може бути промодельована з одним або більшою кількістю векторів входу, які визначаються як початкові умови. Після того, як початкові умови задані, мережа генерує вихід, який по зворотному зв'язку подається на вхід. Цей процес повторюється багато разів, поки вихід не встановиться в положення рівноваги. Можна сподіватися, що кожен вектор виходу зрештою зійдеться до однієї з точок рівноваги, найбільш близької до вхідного сигналу.

Розглянемо наступний приклад. Припустимо, що потрібно розробити мережу із двома стійкими точками у вершинах тривимірного куба:

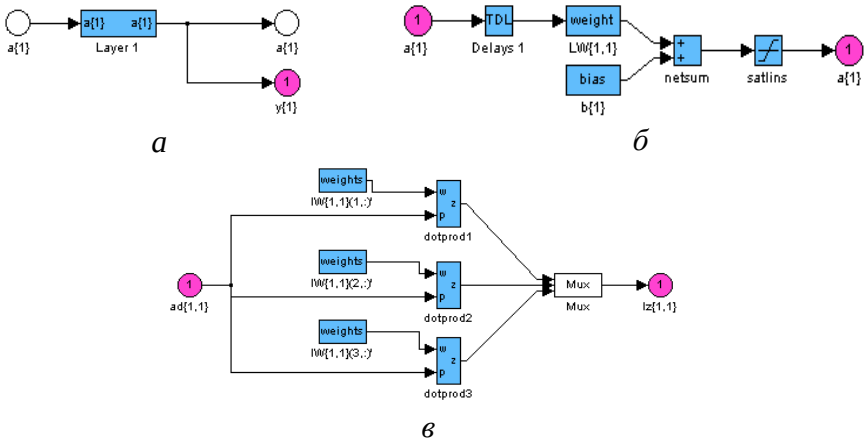
```
T = [-1 -1 1; 1 -1 1]'
T =
-1   1
-1  -1
```

```
 1   1
```

Виконаємо синтез мережі:

```
net = newhop(T);
gensim(net)
```

Структура мережі представлена рис. 6.



а *б* *в*

Рис. 6

Читачеві слід звернути увагу, що у схемі рис. 6,*а* вхід та вихід прошарку збігаються; на рис. 6, *б* показано елементи, які використовуються в рекурентному прошарку. Звернемося до інформації про структуру прошарку:

```
net.layers{1}
ans =
dimensions: 3
distanceFcn: 'dist'
distances: [3×3 double]
initFcn: 'initwb'
netInputFcn: 'netsum'
positions: [0 1 2]
size: 3
topologyFcn: 'hextop'
transferFcn: 'satlins'
userdata: [1×1 struct]
```

З цього списку випливає, що у прошарку використовується функція ініціалізації initwb, функція підсумовування входів netsum та функція активації satlins.

На рис. 6,*в* показано блок, що описує матрицю ваг, яка визначає перехідну матрицю динамічної моделі рекурентного прошарку.

Динамічна модель рекурентного прошарку модифікованої мережі Хопфілда описується так:

$$\mathbf{a}^1(k) = \text{satlins}(\mathbf{LW}^{11}\mathbf{a}^1(k-1) + \mathbf{b}^1), \quad \mathbf{a}^1(0) = \mathbf{p}.$$

При уважному аналізі наведеного співвідношення можна переконатися, що матриця ваг LW^{11} рівносильна перехідній матриці динамічної системи, а вектор зсувів b^1 – вектору передачі одиничного входу. Необхідно сформувати ці елементи, якщо задані точки t рівноваги системи у вершинах гіперкуба.

Синтез мережі

Алгоритм синтезу реалізований у ППП NT у вигляді підфункції solvehop2(t) M-функції newhop.

Якщо встановлено множину цільових точок рівноваги, представлених матрицею T, то функція newhop повертає матрицю ваг і вектор зсувів для рекурентного прошарку мережі

Хопфілда. При цьому гарантується, що точки стійкої рівноваги відповідатимуть цільовим векторам, але можуть з'явитися так звані паразитні точки. У процесі синтезу мережі кількість таких небажаних точок зводиться до мінімуму.

Нехай задано Q цільових векторів, що утворюють матрицю T розміру S×*Q*:

$$\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_{Q-1}, \mathbf{t}_Q].$$

Утворимо нову матрицю Y розміру S×*Q*-1 Наступного виду:

$$\mathbf{Y} = [\mathbf{t}_1 - \mathbf{t}_Q, \mathbf{t}_2 - \mathbf{t}_Q, ..., \mathbf{t}_{Q-1} - \mathbf{t}_Q]$$

Обчислимо розкладання матриці Y за сингулярними числами:

$$\mathbf{Y} = \sum_{i=1}^{S} \sigma_i \mathbf{u}_i \mathbf{v}_i^{\mathrm{T}}.$$

Далі утворюємо матриці

$$\mathbf{TP} = \sum_{i=1}^{K} \mathbf{u}_i \mathbf{u}_i^{\mathrm{T}}, \quad \mathbf{TM} = \sum_{i=K+1}^{S} \mathbf{u}_i \mathbf{u}_i^{\mathrm{T}},$$

де K – ранг матриці Y.

Побудуємо динамічну систему

$$\dot{\mathbf{x}} = \mathbf{T}_\tau \mathbf{x} + (\mathbf{E} - \mathbf{T}_\tau)\mathbf{t}_{\mathrm{Q}},$$

де T_τ– матриця виду TP –τTM;τ- Динамічний параметр (в алгоритмі прийнятий рівним 10);**E**-одинична матриця.

Обчислимо дискретну модель попередньої системи:

$$\mathbf{x}_k = \Phi \mathbf{x}_{k-1} + \mathbf{F}\mathbf{t}_{\mathrm{Q}},$$

де Ф – перехідна матриця виду $e^{\mathbf{T}_\tau h}$; F – матриця передачі входу виду $\mathbf{F} = (c_1\mathbf{TP} + c_2\mathbf{TM})(\mathrm{E} - \mathbf{T}_\tau)$, $c_1 = e^h - 1$, $c_2 = (1-e^{-th})/\tau$; h – такт дискретності (в алгоритмі прийнято рівним 0.15 с).

Відповідна матриця ваг рекурентного прошарку та вектор зміщення обчислюються наступним чином:

$$\begin{cases} \mathbf{W} = \mathbf{\Phi}; \\ \mathbf{b} = \mathbf{F}\mathbf{t}_{\mathrm{Q}}. \end{cases}$$

Коли мережа спроектована, вона може бути перевірена одним або більше числом векторів входу. Цілком імовірно, що вектори входу, близькі до цільових точок рівноваги, знайдуть свої цілі. Здатність модифікованої мережі Хопфілда швидко обробляти набори векторів входу дозволяє перевірити мережу за відносно короткий час. Спочатку можна перевірити, що точки рівноваги цільових векторів дійсно належать вершинам гіперкуба, а потім можна визначити області тяжіння цих точок і виявити паразитні точки рівноваги, якщо вони є.

Розглянемо наступний приклад. Припустимо, що потрібно створити модифіковану мережу Хопфілда з двома точками рівноваги, заданими у тривимірному просторі:

```
T = [-1 -1 1; 1 -1 1]'
T =
-1   1
-1  -1
 1   1
```

Виконаємо синтез мережі, використовуючи M-функцію newhop:

```
net = newhop(T);
```

Упевнимося, що розроблена мережа має стійкі стани у цих двох точках. Виконаємо моделювання мережі Хопфілда, врахувавши, що ця мережа не має входів і містить рекурентний прошарок; у цьому випадку цільові вектори визначають початковий стан прошарку Ai, а другий аргумент функції sim - це число цільових векторів:

```
Ai = T;
[Y, Pf, Af] = sim (net, 2, [], Ai);
Y
Y =
-1  1
-1 -1
 1  1
```

Справді, стійкі положення рівноваги мережі перебувають у визначених точках.

Задамо іншу початкову умову у вигляді масиву комірок:

```
Ai = {[-0.5; 0.6; -0.7]};
Ai {1,1}
ans =
-0.5000
 0.6000
-0.7000
```

Ця точка розташована поблизу першого положення рівноваги, так що можна очікувати, що мережа буде сходитися саме до цієї точки. Щоб виконати моделювання мережі необхідно вхідний вектор перетворити в послідовний формат та вказати його як початкові умови лінії затримки. Оскільки процес генерування вихідних сигналів мережі, які за зворотним зв'язком подаються на вхід, повторюється багато разів, то при застосуванні функції sim в якості другого аргументу необхідно вказати послідовність з двох значень, перше – це такт дискретності (визначає інтервал між двома послідовними моментами знімання даних), друге – кількість кроків моделювання (кількість ітерацій), а в якості четвертого аргументу – початкові умови лінії затримки. Тоді результатом буде масив комірок, що містить послідовні значення виходів мережі, які формуються ітераційно. Для перегляду отриманих результатів дані необхідно перетворити у звичайний масив за допомогою функції cat:

```
[Y, Pf, Af] = sim (net, {1 5}, {}, Ai);
Y=cat(2, Y{:})
Y =
-0.5809   -0.6749   -0.7842   -0.9111   -1.0000
-0.7207   -1.0000   -1.0000   -1.0000   -1.0000
 0.6984    1.0000    1.0000    1.0000    1.0000
```

Справді, із заданого початкового стану мережа повернулася у стійке положення рівноваги. Бажано, щоб мережа поводилася аналогічним чином при заданні будь-якої початкової точки в межах куба, вершини якого складені з усіх комбінацій чисел 1 і -1 в тривимірному просторі. На жаль, цього не можна гарантувати і досить часто мережі Хопфілда включають небажані паразитні точки рівноваги.

Починаючи роботу в стані Ai, мережа може потрапити в одне з наступних трьох положень:

– прийти у стійкий стан, що відповідає еталонному образу, який є найбільш близьким до Ai;

– прийти у стійкий стан, що не відповідає ніякому еталону (паразитний), тобто до помилкового образу;

– виявитися втягнутою у коливальний процес.

Приклад:

Розглянемо мережу Хопфілда з чотирма нейронами та визначимо 4 точки рівноваги у вигляді наступного масиву цільових векторів:

```
T = [1 -1; -1 1; 1 1; -1 -1]'
T =
 1-1 1 -1
-1 1 1 -1
```

На рис. 7 показано ці 4 точки рівноваги на площині станів мережі Хопфілда.

```
plot(T(1,:), T(2,:),'*r') % Рис.7
axis([-1.1 1.1 -1.1 1.1])
title('Точки рівноваги мережі Хопфілда')
xlabel('a(1)'), ylabel('a(2)')
```

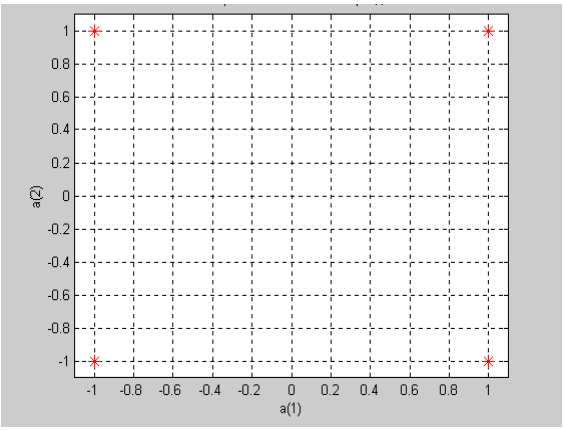


Рис. 7

Розрахуємо ваги та зміщення модифікованої мережі Хопфілда, використовуючи М-функцію newhop:

```
net = newhop(T);
W = net.LW {1,1}
b = net.b {1,1}
W =
1.1618 0
0 1.1618
b =
3.5934e-017
3.5934e-017
```

Перевіримо, чи належать вершини квадрата до мережі Хопфілда:

```
Ai = T;
[Y,Pf,Af] = sim(net,4,[],Ai)
Y =
   1  -1   1  -1
  -1   1   1  -1
Pf = []
Af =
   1  -1   1  -1
  -1   1   1  -1
```

Як і слід очікувати, виходи мережі дорівнюють цільовим векторам.

Тепер перевіримо поведінку мережі за випадкових початкових умов:

```
plot(T(1,:), T(2,:),'*r'), hold on
axis([-1.1 1.1 -1.1 1.1])
xlabel('a(1)'), ylabel('a(2)')
new = newhop(T);
[Y, Pf, Af] = sim (net, 4, [], T);
```

```
for i =1:25
a = {rands(2,1)};
[Y, Pf, Af] = sim (net, {1,20}, {}, a);
record = [cell2mat(a) cell2mat(Y)];
start = cell2mat(a);
plot(start(1,1), start(2,1), 'kx', record(1,:), record(2,:))
end
```

Результат подано на рис. 8.

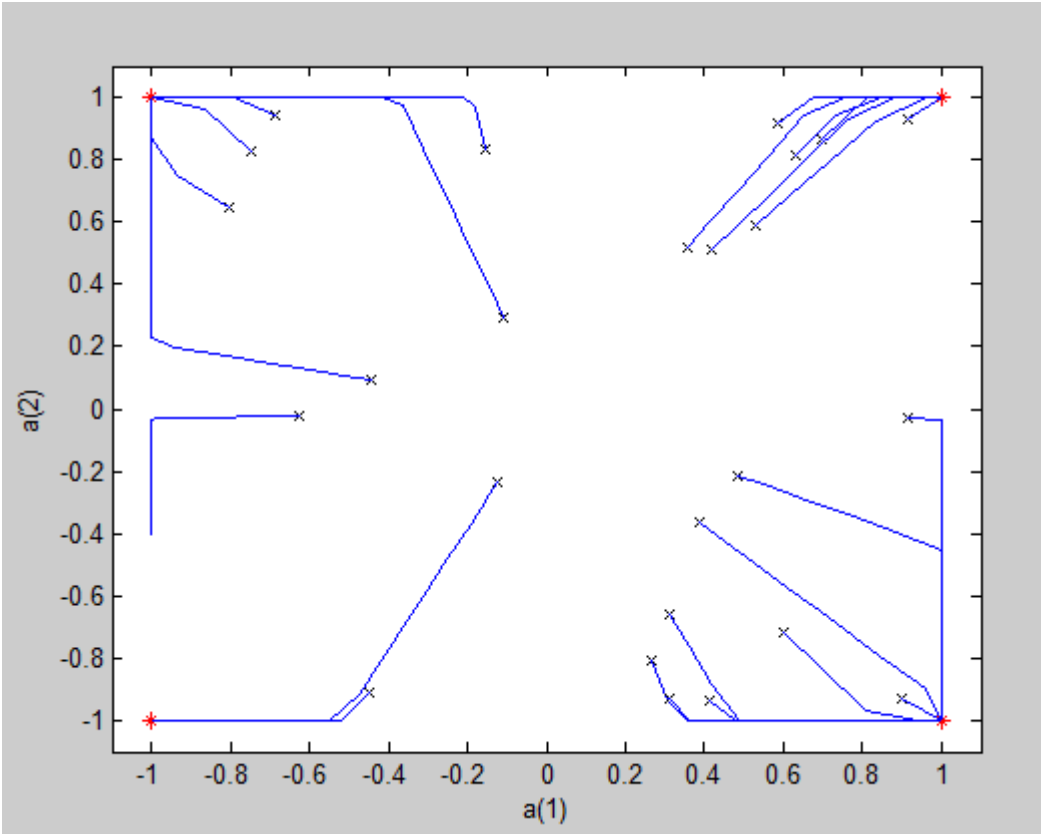


Рис. 8

Читач може продовжити вивчення модифікованих мереж Хопфілда, звернувшись до демонстраційних прикладів. Приклад двовимірної мережі можна знайти у демонстрації demohop1, приклад нестійкої точки рівноваги – демонстрації demohop2. Демонстрація demohop3 показує приклад тривимірної модифікованої мережі Хопфілда. Демонстрація demohop4 ілюструє появу стійких паразитних точок рівноваги.