Міністерство освіти і науки України
Кіровоградський національний технічний університет
Кафедра програмування та захисту інформації

Комп'ютерні системи штучного інтелекту

Методичні вказівки
до виконання лабораторних робіт студентами
денної та заочної форми навчання спеціальностей
123 "Комп’ютерна інженерія" та
122 "Комп’ютерні науки та інформаційні технології"

Кіровоград 2016

Міністерство освіти і науки України
Кіровоградський національний технічний університет
Кафедра програмування та захисту інформації

Мелешко Є.В.

Комп'ютерні системи штучного інтелекту

Методичні вказівки
до виконання лабораторних робіт студентами
денної та заочної форми навчання спеціальностей
123 "Комп’ютерна інженерія" та
122 "Комп’ютерні науки та інформаційні технології"

Затверджено
на засіданні кафедри
програмування та
захисту інформації
Протокол №13
від 15.02.16

Кіровоград 2016

Комп'ютерні системи штучного інтелекту. Методичні вказівки до виконання лабораторних робіт студентами денної та заочної форми навчання спеціальностей 123 "Комп’ютерна інженерія", 122 "Комп’ютерні науки та інформаційні технології" / Укл.: Є.В. Мелешко – Кіровоград: КНТУ, 2016. – 61 с.

Дані методичні вказівки адресовані майбутнім розробникам програмного забезпечення та фахівцям з аналізу даних, також можуть використовуватися студентами інших спеціальностей. Вони включають в себе основи проектування комп’ютерних систем штучного інтелекту, містять основні теоретичні положення та практичні завдання, необхідні для засвоєння навчального матеріалу.

Укладач: Мелешко Є.В., канд. техн. наук, доцент

Рецензенти: Смірнов О.А., доктор техн. наук, професор
Якименко Н.М., канд. фіз.-мат. наук, доцент

ЗМІСТ

Лабораторна робота №1

Тема: Штучні нейронні мережі. Моделювання формальних логічних функцій. Прогнозування часових рядів

Мета: Отримати початкові навички по створенню штучних нейронних мереж, що здатні виконувати прості логічні функції, та нейронних мереж, що здатні прогнозувати часові ряди.

Теоретичні відомості

Штучний нейрон – вузол штучної нейронної мережі, що є спрощеною моделлю природного нейрона.

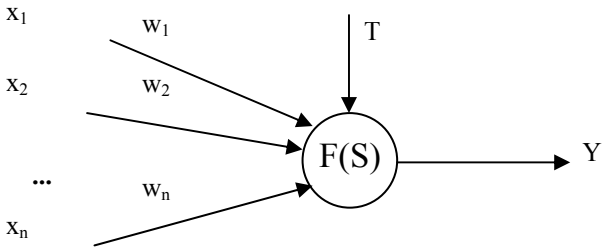


Рисунок 1.1 – Схема штучного нейрона

x_1- x_n – входи нейрона (синапси);

w_1- w_n – вагові коефіцієнти входів;

S – зважена сума входів нейрона;

F(S) – функція активації нейрона;

T – порогове значення (значення, після якого нейрон переходить у стан збудження), є не у всіх типів штучних нейронів;

Y – вихід нейрона (аксон).

Зважена сума S обчислюється за наступною формулою:

$$S = x_1 \cdot w_1 + x_2 \cdot w_2 + ... + x_n \cdot w_n. \quad (1.1)$$

Функція активації F(S) – визначає залежність сигналу на виході нейрона від зваженої суми сигналів на його входах. Використання різних функцій активації дозволяє вносити нелінійність в роботу нейрона і в цілому нейронної мережі.

Приклади функцій активації

Лінійна передавальна функція:

$$F(S) = \begin{cases} 0 \ if \ S \le 0 \\ 1 \ if \ S \ge 1 \\ S \ else \end{cases}, \quad (1.2)$$

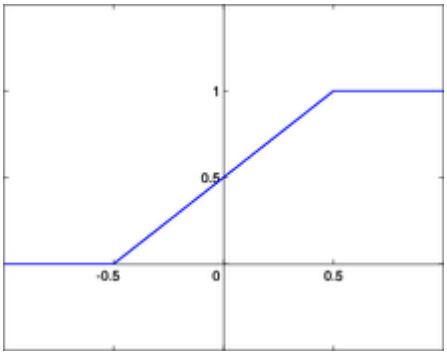


Рисунок 1.2 – Лінійна передавальна функція

Порогова передавальна функція:

$$F(S) = \begin{cases} 1 \ if \ S \geq 0 \\ 0 \ else \end{cases}, \tag{1.3}$$

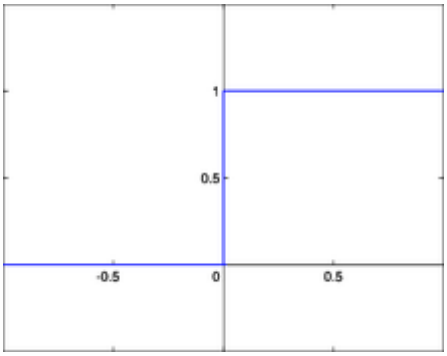


Рисунок 1.3 – Порогова передавальна функція

Сигмоїдальна передавальна функція:

$$F(S) = \frac{1}{(1 + \exp(-S))}, \tag{1.4}$$

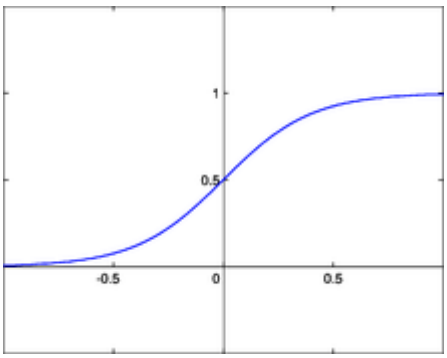


Рисунок 1.4 – Сигмоїдальна передавальна функція

Моделювання формальних логічних функцій за допомогою нейронів та нейронних мереж

Моделювання логічної функції "I" (AND)

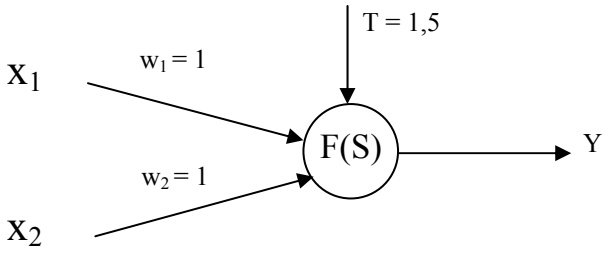


Рисунок 1.5 – Схема штучного нейрону, налаштованого на моделювання логічної функції "I"

Функція активації даного нейрона:

$$F(S) = \begin{cases} 0 \ if \ S < 1{,}5 \\ 1 \ if \ S >= 1{,}5 \end{cases} \quad (1.5)$$

Таблиця 1.1 – Таблиця істинності логічної функції "I" (AND)

x_1	x_2	Y
0	0	0
0	1	0
1	0	0
1	1	1

Розглянемо як обчислюється вихідний сигнал даного нейрона при різних вхідних даних:

$x_1 = 0$; $x_2 = 0$
$S = 0 \cdot 1 + 0 \cdot 1 = 0$
$Y = F(S) = 0$ (тому що $S < 1{,}5$)

$x_1 = 0$; $x_2 = 1$
$S = 0 \cdot 1 + 1 \cdot 1 = 1$
$Y = F(S) = 0$ (тому що $S < 1{,}5$)

$x_1 = 1$; $x_2 = 0$
$S = 1 \cdot 1 + 0 \cdot 1 = 1$
$Y = F(S) = 0$ (тому що $S < 1{,}5$)

$x_1 = 1$; $x_2 = 1$
$S = 1 \cdot 1 + 1 \cdot 1 = 2$
$Y = F(S) = 1$ (тому що $S > 1{,}5$)

Моделювання логічної функції "АБО" (OR)

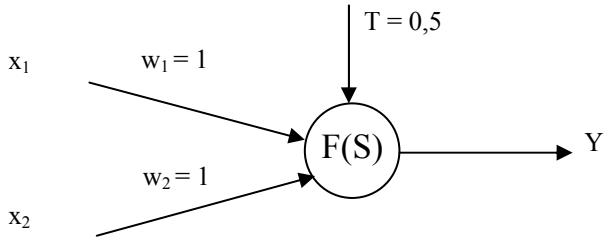


Рисунок 1.6 – Схема штучного нейрону, налаштованого на моделювання логічної функції "АБО"

Функція активації даного нейрона:

$$F(S) = \begin{cases} 0 \ \ f \ \ S < 0{,}5 \\ 1 \ \ f \ \ S >= 0{,}5 \end{cases} \qquad (1.6)$$

Таблиця 1.2 – Таблиця істинності логічної функції "АБО" (OR)

x_1	x_2	Y
0	0	0
0	1	1
1	0	1
1	1	1

Розглянемо як обчислюється вихідний сигнал даного нейрона при різних вхідних даних:

$x_1 = 0$; $x_2 = 0$
$S = 0 \cdot 1 + 0 \cdot 1 = 0$
$Y = F(S) = 0$ (тому що $S < 0{,}5$)

$x_1 = 0$; $x_2 = 1$
$S = 0 \cdot 1 + 1 \cdot 1 = 1$
$Y = F(S) = 1$ (тому що $S > 0{,}5$)

$x_1 = 1$; $x_2 = 0$
$S = 1 \cdot 1 + 0 \cdot 1 = 1$
$Y = F(S) = 1$ (тому що $S > 0{,}5$)

$x_1 = 1$; $x_2 = 1$
$S = 1 \cdot 1 + 1 \cdot 1 = 2$
$Y = F(S) = 1$ (тому що $S > 0{,}5$)

Моделювання логічної функції "НІ" (NOT)

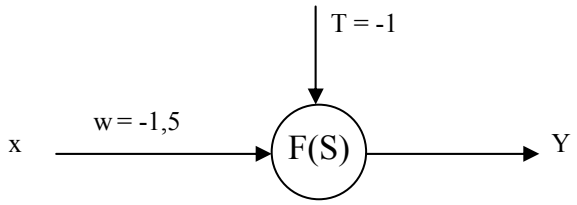


Рисунок 1.7 – Схема штучного нейрону, налаштованого на моделювання логічної функції "НІ"

Функція активації даного нейрона:

$$F(S) = \begin{cases} 0 \ if \ S < -1 \\ 1 \ if \ S >= -1 \end{cases} \quad (1.7)$$

Таблиця 1.3 – Таблиця істинності логічної функції "НІ" (NOT)

x	Y
0	1
1	0

Розглянемо як обчислюється вихідний сигнал даного нейрона при різних вхідних даних:

x = 0
S = 0·(-1,5) = 0
Y = F(S) = 1 (тому що S > -1)

x = 1
S = 1·(-1,5) = -1,5
Y = F(S) = 0 (тому що S < -1)

Моделювання логічної функції "Виключне АБО" (XOR)

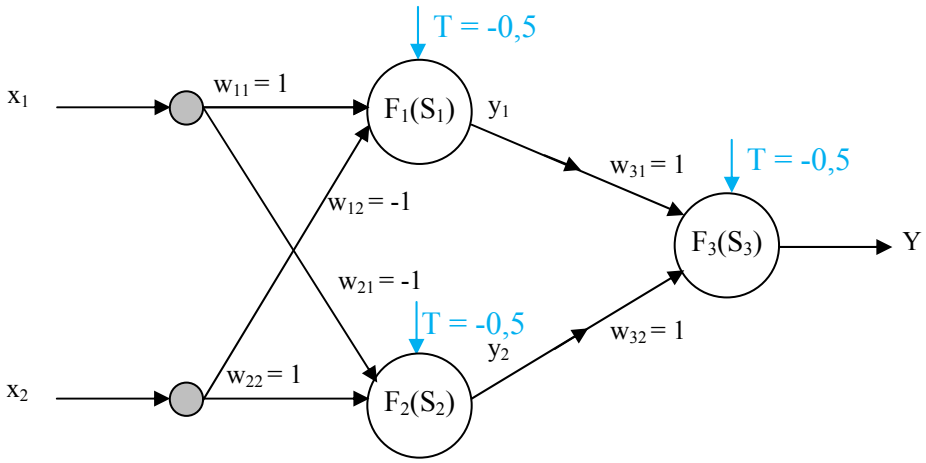


Рисунок 1.8 – Схема штучної нейронної мережі, налаштованої на моделювання логічної функції "Виключне АБО" (XOR)

Таблиця 1.4 – Таблиця істинності логічної функції "Виключне АБО" (XOR)

x_1	x_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

Функція активації даного нейрона:

$$F(S) = \begin{cases} 0 \ if \ S < 0,5) \\ 1 \ if \ S >= 0,5 \end{cases} \quad (1.8)$$

Розглянемо як обчислюється вихідний сигнал даної мережі при різних вхідних даних:

$x_1 = 1$; $x_2 = 1$
$S_1 = 1 \cdot 1 + 1 \cdot (-1) = 0$
$Y_1 = F(S) = 0$ (тому що $S < 0,5$)
$S_2 = 1 \cdot (-1) + 1 \cdot 1 = 0$
$Y_2 = F(S) = 0$ (тому що $S < 0,5$)
$S_3 = 0 \cdot 1 + 0 \cdot 1 = 0$
$Y_3 = F(S) = 0$ (тому що $S < 0,5$)

$x_1 = 0$; $x_2 = 1$
$S_1 = 0 \cdot 1 + 1 \cdot (-1) = -1$
$Y_1 = F(S) = 0$ (тому що $S < 0,5$)
$S_2 = 0 \cdot (-1) + 1 \cdot 1 = 1$
$Y_2 = F(S) = 1$ (тому що $S > 0,5$)
$S_3 = 0 \cdot 1 + 1 \cdot 1 = 1$
$Y_3 = F(S) = 1$ (тому що $S > 0,5$)

Прогнозування часових рядів за допомогою нейрону з сигмоїдальною функцією активації

Задача прогнозування часових рядів, в яких є певні закономірності, може бути вирішена за допомогою нейромережі, яка може навчатися. Відомо, що людський мозок здатний до самонавчання, причому досягає успіхів найчастіше, не знаючи природи процесів, що лежать в основі виконуваних дій.

Наприклад, щоб потрапити м'ячем у баскетбольне кільце, робот-баскетболіст повинен виміряти відстань до кільця й напрямок, розрахувати параболічну траєкторію, і зробити кидок з урахуванням маси м'яча й опору повітря. Людина ж обходиться без цього тільки через тренування. Багаторазово здійснюючи кидки й спостерігаючи результати, вона коректує свої дії, поступово вдосконалюючи свою техніку. При цьому в її мозку формуються

відповідні структури нейронів, відповідальні за техніку кидків.

Рисунок 1.9 – Алгоритм навчання нейронних мереж

1. Вибір структури нейромережі, це складна задача, яку ми будемо розглядати в наступних лабораторних роботах. В даній лабораторній роботі візьмемо мережу, що складається з одного нейрону, зображеного на рисунку 1.

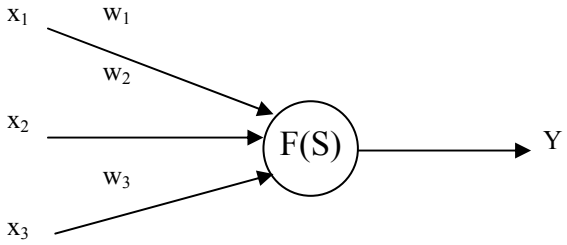


Рисунок 1.10 – Штучний нейрон для прогнозування значень часового ряду

Зважена сума та функція активації даного нейрона:

$$S_i = x_{i-3} \cdot w_1 + x_{i-2} \cdot w_2 + x_{i-1} \cdot w_3, \quad (1.9)$$

$$Y_i = 1/(1+\exp(-S_i))*10, \quad (1.10)$$

де w_1, w_2, w_3 – синаптичні ваги;

x_{i-3}, x_{i-2}, x_{i-1} – вхідні сигнали – відомі попередні значення часового ряду (*i-й* набор вхідних даних);

S_i – зважена сума *i-го* набору вхідних даних;

Y_i – прогнозоване значення *i-го* члена часового ряду x_i;

10 – масштабний множник.

2. Навчання полягає в тому, що на вхід мережі подаються спеціальні тренувальні дані, тобто такі вхідні дані, вихідний результат для яких відомий. На виході формуються результуючі дані, результати порівнюються з

очікуваними, і обчислюється значення помилки. Після цього в певній послідовності виконується корекція параметрів нейронної мережі із метою мінімізації функції помилки. Якщо задовільної точності досягти не вдається, варто змінити структуру мережі й повторити навчання на множині тренувальних даних.

Таблиця 1.5 – Приклад тренувальних даних для нейронної мережі, що здійснює прогнозування значень часового ряду

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
1,59	5,73	0,48	5,28	1,35	5,91	0,77	5,25	1,37	4,42	0,26	4,21	1,90	4,08	1,40

Перші 13 чисел будемо використовувати для навчання мережі як тренувальний набір даних. Останні два члени ряду в навчанні не будуть брати участь, а служитимуть для тестування мережі.

Прогнозування полягає в тому, щоб на основі x_i, x_{i+1}, x_{i+2} обчислити x_{i+3}. Іншими словами, нейронна мережа «ковзає» уздовж часового ряду, «обмацуючи» синапсами по три сусідніх числа, та намагається прогнозувати значення наступні за ними. Таким чином, для наведеного вище прикладу вхідними й вихідними величинами будуть наступні (див. табл. 1.6).

Таблиця 1.6 – Очікувані значення часового ряду на кожному кроці навчання

і-тий набір даних	**Вхід нейрона**	**Вихід (очікуваний результат)**
1	1,59 5,73 0,48	5,28
2	5,73 0,48 5,28	1,35
3	0,48 5,28 1,35	5,91
4	5,28 1,35 5,91	0,77
	і т.д.	

Навчання нейронної мережі полягає в знаходженні таких значень ваг w, при яких нейромережа буде здатна видавати на основі вхідних даних вірні вихідні дані з певною наперед заданою точністю.

Дана задача задовільно вирішується за допомогою **алгоритму зворотного поширення (*back propagation*)**, що полягає в наступному:

1) Спочатку всі вагові коефіцієнти нейронної мережі встановлюються довільно. Можна скористатися функцією random, або просто присвоїти всім ваговим коефіцієнтам 1.

2) Через мережу пропускаються тренувальні дані (перший набір вхідних даних), і обчислюється сумарна функція помилки (сума квадратів помилки):

$$E = \sum_{i=1}^{N} (Y_i - y_i)^2, \qquad (1.11)$$

де Y_i – обчислене значення виходу нейрона;

y_i – правильне значення наступного члену часового ряду.

3) Обчислюється значення похідної функції помилки E'_t для кожного вагового коефіцієнта:

$$E'_i = (Y_i - y_i)\cdot(exp(-s_i) / (1+exp(-s_i))^2)\cdot x_i \quad (1.12)$$

4) На основі E'_{i1}, E'_{i2}, та E'_{i3}, здійснюється розрахунок виправлень Δw_{i1}, Δw_{i2}, та Δw_{i3} до відповідних вагових коефіцієнтів за наступною формулою:

$$\Delta w_i = - v \cdot E'_i, \quad (1.13)$$

де v – коефіцієнт швидкості навчання. Виправлення необхідно знайти для кожного *і-го* набору вихідних даних, і обчислити середні значення $\Delta w_{середнє1}$, $\Delta w_{середнє2}$, та $\Delta w_{середнє3}$ для всього набору:

$$\Delta w_{\text{середнє}} = \frac{1}{N}\sum_{i=1}^{N}\Delta w_i \ , \quad (1.14)$$

де N – кількість наборів вхідних даних для навчання.

5) Вагові коефіцієнти коректуються на величину обчислених виправлень:

$$w = w + \Delta w_{середнє}, \quad (1.15)$$

6) Поточне значення сумарної функції помилки E зберігається в іншій змінній:

$$E_0 = E. \quad (1.16)$$

Кроки 2–5 алгоритму зворотного поширення повторюються для кожного *і-того* набору вхідних даних (назвемо це *цикли навчання*), поки функція помилки не знизиться до заданого рівня, наприклад:

$$|E - E_0| < 0{,}0001 \quad (1.17)$$

Кількість ітерацій у процесі навчання мережі може досягати сотень і навіть тисяч. Тому доречно зробити додаткову умову виходу із циклу, на випадок якщо навчання з заданим рівнем точності буде тривати непримустимо довго, або відбудеться зациклення. Додатковою умовою виходу може бути натиснення користувачем кнопки "Стоп", або досягнення певної кількості циклів навчання, наприклад: i > 1000000.

3. Тестування, тобто контроль точності на спеціальних тестових даних, виконується після того, як нейронна мережа навчена. Це означає, що всі дані варто розбити на дві підмножини: на першій з них виконується навчання мережі, а на другій - тестування. За аналогією з навчанням людини тестування можна вподібнити іспиту. В нашому випадку для тестових даних ми залишили визначення нейронною мережею чисел x_{14} та x_{15} нашого часового ряду.

Завдання:

Написати програму для реалізації штучних нейронів та нейронних мереж для:

- моделювання логічної функції І,

- моделювання логічної функції АБО,
- моделювання логічної функції НІ,
- моделювання логічної функції Виключне АБО,
- прогнозування часового ряду (приклади часових рядів див. в додатку до лабораторної роботи А).

Додаткове завдання:

Написати програму для реалізації штучної нейронної мережі, що моделює логічну функцію, таблиця істинності якої наводиться в таблиці 1.7.

Таблиця 1.7 – Таблиця істинності логічної функції

x_1	x_2	x_2	y
0	0	0	1
0	1	0	1
1	0	0	0
1	1	1	1

Контрольні питання:

1. З яких елементів складається штучний нейрон? Яке призначення цих елементів?
2. Що таке функія активіції? Які інують функції активації?
3. Як можна змоделювати за допомогою штучного нейрона логічну функцію AND?
4. Як можна змоделювати за допомогою штучного нейрона логічну функцію OR?
5. Як можна змоделювати за допомогою штучного нейрона логічну функцію NOT?
6. З яких основних блоків складається алгоритм навчання нейронних мереж? Як ці блоки пов’язані між собою?
7. В чому заключається алгоритм зворотного поширення помилки?
8. Яким чином можна налаштувати нейронну мережу на прогнозування значень величин часового ряду?
9. Яку структуру має нейронна мережа, що моделює логічну функцію XOR?

Додаток А1

Таблиця А1.1 - Варіанти часових рядів

№ варіанту	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
1	2,56	4,20	1,60	4,29	1,17	4,40	0,88	4,14	0,07	4,77	1,95	4,18	0,04	5,05	1,40
2	0,20	5,14	0,47	4,37	1,22	4,29	1,89	4,51	0,32	5,80	1,37	5,77	0,88	4,86	1,94
3	1,92	4,01	1,48	5,45	1,56	5,42	1,28	4,34	1,51	5,49	1,32	4,00	0,49	4,19	1,53
4	0,13	5,97	0,57	4,02	0,31	5,55	0,15	4,54	0,65	4,34	1,54	4,70	0,58	5,83	0,03
5	2,16	3,19	1,85	4,84	0,55	4,20	1,68	4,74	0,14	5,68	0,48	5,03	0,18	5,99	0,09
6	2,54	5,28	0,78	5,72	0,58	4,65	0,91	5,80	1,76	5,67	1,73	5,70	1,03	5,00	1,79
7	1,69	3,38	1,40	5,56	1,86	5,62	0,46	5,51	0,26	5,13	1,18	5,98	1,36	5,09	1,29

№ варіанту	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
8	1,19	5,61	0,89	6,00	1,04	5,98	0,03	6,00	1,83	4,23	0,60	4,15	0,13	5,01	1,87
9	0,87	4,12	0,93	4,62	1,51	5,76	0,50	5,48	0,95	4,03	0,92	5,15	1,66	5,01	0,40
10	2,82	3,48	0,60	4,76	1,51	5,51	1,48	5,19	0,48	5,22	0,21	4,19	0,07	4,63	0,49
11	2,64	4,66	1,87	4,05	1,73	5,31	1,67	5,96	0,13	5,64	1,52	4,07	0,22	4,79	0,73
12	2,65	5,60	1,21	5,48	0,73	4,08	1,88	5,31	0,78	4,36	1,71	5,62	0,43	4,21	1,21
13	2,37	4,85	1,97	4,17	1,39	4,66	1,26	4,40	0,46	5,54	1,34	5,80	1,61	5,97	1,95
14	1,88	4,52	1,91	5,66	1,23	5,50	1,14	5,29	1,60	4,31	0,06	5,33	0,07	4,62	0,69
15	0,78	4,95	1,19	4,08	0,80	4,25	0,22	4,63	1,48	4,97	0,53	5,50	1,28	5,79	0,44
16	0,58	3,38	0,91	5,80	0,91	5,01	1,17	4,67	0,60	4,81	0,53	4,75	1,01	5,04	1,07
17	0,51	4,82	0,43	4,71	1,92	5,86	1,24	4,69	0,72	5,26	0,90	4,55	1,46	5,21	1,50
18	0,07	3,58	0,44	5,33	0,56	5,24	1,99	4,38	0,89	4,53	1,82	4,13	1,88	5,97	1,18
19	1,44	4,60	1,22	5,90	1,34	4,31	1,02	4,35	0,82	4,18	1,60	4,86	1,45	4,97	1,00
20	2,57	4,35	1,27	5,46	1,30	4,92	1,31	4,14	1,97	5,67	0,92	4,76	1,72	4,44	1,49
21	0,79	3,84	0,92	4,50	0,96	5,51	1,14	5,32	0,39	4,99	1,36	5,81	1,90	4,79	1,41
22	0,99	4,72	1,59	5,29	1,53	5,58	0,84	5,79	0,21	5,94	0,42	5,98	1,18	5,55	0,11
23	2,92	3,56	0,15	5,11	1,38	4,44	1,61	4,11	1,97	4,50	1,37	5,08	1,76	5,19	1,58
24	0,48	4,30	0,91	4,85	0,53	4,51	1,95	5,88	0,63	5,79	0,92	5,18	1,88	4,84	0,22
25	1,88	4,98	0,06	5,26	1,16	5,06	0,58	5,28	1,41	5,57	1,19	5,36	1,40	4,30	0,09
26	2,57	5,77	0,38	4,73	0,10	5,93	1,35	4,70	1,62	5,51	1,78	5,66	1,47	5,52	1,88
27	0,11	4,87	1,52	4,47	0,34	5,44	1,20	5,21	1,48	5,93	0,62	5,48	1,34	4,25	0,65
28	1,07	3,17	1,08	5,99	1,28	4,11	0,25	5,82	0,96	4,83	1,10	4,31	0,81	5,49	1,92
29	1,59	5,74	0,48	5,28	1,34	5,91	0,77	5,25	1,37	4,42	0,26	4,21	1,90	4,08	1,40
30	0,68	5,78	0,25	5,58	1,31	4,28	1,57	5,75	0,41	5,55	0,90	5,86	0,03	5,57	0,30

Лабораторна робота №2

Тема: Моделювання нейронних мереж у візуальному середовищі fannExplorer відкритої бібліотеки fann

Мета: Ознайомитися з відкритою бібліотекою fann для розробки штучних нейронних мереж та її графічним середовищем fannExplorer

Теоретичні відомості

FANN (Fast Artificial Neural Network) – відкрите програмне забезпечення для побудови штучних нейронних мереж. У даного рішення є API для 15 мов програмування, що робить можливим використовувати його у різних середовищах програмування.

Сайт бібліотеки FANN розташований за наступним посиланням: http://leenissen.dk/

Завантажити бібліотеку та декілька варіантів графічної оболонки для неї можна за посиланням http://leenissen.dk/fann/wp/download/

Розглянемо графічну оболонку fannExplorer, її разом з документацією до неї можна завантажити за посиланням http://leenissen.dk/fann/gui.php.

Після завантаження архіву fannExplorer21 розпакуйте його та зайдіть у підкаталог fann\fannSoap\fannKernel\Release, в якому запустіть файл fannKernel.exe.

З’явиться наступне вікно зображене на рис. 2.1.

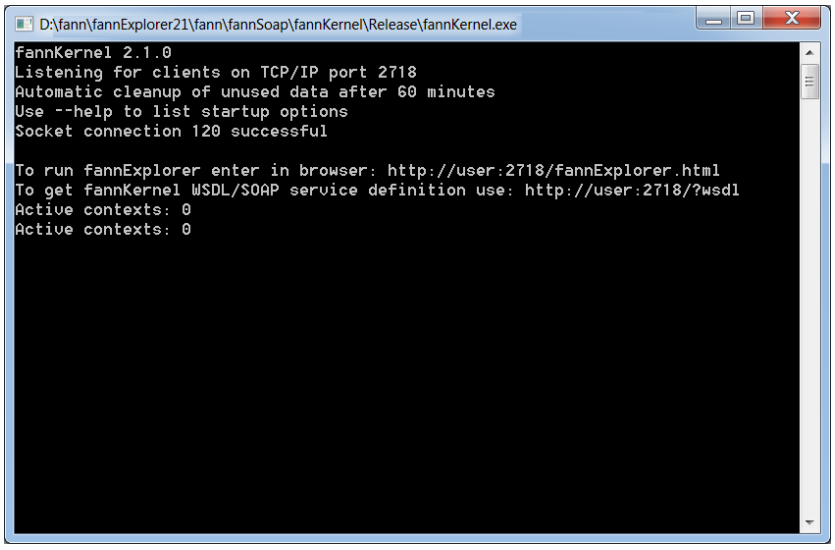


Рисунок 2.1 – Вікно додатку fannKernel.exe після його запуску

Залишивши запущеним додаток fannKernel.exe, наберіть у інтернет-браузері наступну адресу: http://user:2718/fannExplorer.html

На екрані з’явиться вікно графічного інтерфейсу для бібліотеки fann (рис. 2.2).

Рисунок 2 – Стартова сторінка програми fannExplorer

Для початку роботи з програмою слід вибрати пункт меню користувача View та відмітити прапорцями його підпункти Controller, Error Plot, Weight Graph, після чого на екрані з'являться відповідні вікна. Перше вікно слугує для налагодження та проведення моделювання нейронної мережі, два інші - для виведення результатів моделювання в графічній та цифровій формах. За допомогою пункту меню File можна створювати нейронні мережі (New Neural Network), відкривати раніше створені (Load Neural Network), та зберігати нові (Save Neural Network).

Інтерфейс програми fannExplorer виглядає наступним чином (рис. 2.3):

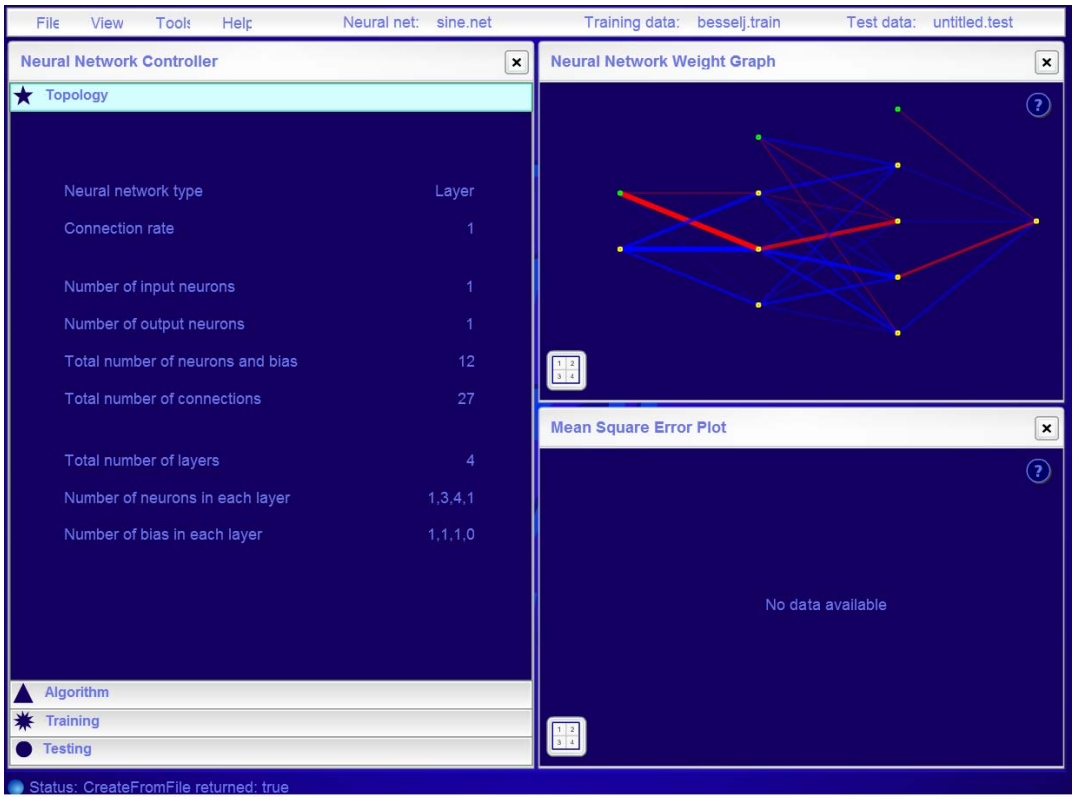


Рисунок 2.3 – Інтерфейс програми fannExplorer

Завдання:

1) Завантажити та запустити програму fannExplorer, дослідити наявні в ній приклади нейронних мереж, які можна переглянути за допомогою пункту меню користувача File->Load Neural Network...

Наприклад для дослідження нейронної мережі xor.net слід:

- відкрити файл xor.net з параметрами нейронної мережі через пункт меню File->Load Neural Network...

- відкрити файл xor.train з навчальною вибіркою даних через пункт меню File->Load Training Data...

- у вікні Neural Network Controller обрати вкладку Training та натиснути кнопку Animate.

- відкрити файл xor.test з тестовою вибіркою даних через пункт меню File->Load Test Data...

- у вікні Neural Network Controller обрати вкладку Testing та натиснути кнопку Execute.

- переглянути одержані результати.

Наведіть у звіті до лабораторної роботи скриншоти дослідженої нейронної мережі xor.net та результатів її роботи. Коротко опишіть як працює нейронна мережа, що обчислює логічну функцію XOR, користуючись довідковою літературою, електронними енциклопедіями та результатами роботи розглянутого прикладу в емуляторі fannExplorer.

2) За посиланням

https://grey.colorado.edu/emergent/index.php/Comparison_of_Neural_Network_Simulators

розташована таблиця, що містить порівняння різних емуляторів нейронних мереж. Виберіть один з емуляторів (серед тих, що поширюються безкоштовно, див. поле License даної таблиці), встановіть та ознайомтеся з його функціями та можливостями. Наведіть у звіті до лабораторної роботи скриншоти дослідженого емулятора та короткий опис його інтерфейсу і можливостей по моделюванню нейронних мереж.

Контрольні питання:

1. Що собою представляє бібліотека FANN? Для чого її можна використовувати?
2. Для чого призначене її розширення fannExplorer? Як його встановити fannExplorer?
3. Як змоделювати в fannExplorer нейронну мережу?
4. В якому вигляді в fannExplorer ввиволятся результати роботи нейронної мережі?

Лабораторна робота №3

Тема: Розпізнавання образів за допомогою штучних нейронних мереж

Мета: Ознайомитися з методами розпізнавання образів за допомогою штучних нейронних мереж, у середовищі fannExplorer побудувати, навчити та протестувати НС для розпізнавання букв

Теоретичні відомості

Штучні нейронні мережі, що використовуються для розпізнавання образів та зображень, моделюють роботу біологічних систем зору.

Розглянемо принципи роботи таких нейронних мереж. На рисунку 1 потоки інформації йдуть зверху донизу. *Сенсорні нейрони*, зображені як очні яблука у верхньому шарі, одержують сигнали із зовнішнього світу й кодують їх у зручну для читання в рамках даної системи форму. У біологічних нейронних мережах (людському мозку) це означає перетворення сигналу в електричні імпульси, у штучних нейронних мережах, які моделюють мозок, - оцифровку сигналу. Кодовані в такий спосіб сигнали передаються нейронам у наступний шар, розташований нижче. *Ефекторні нейрони,* зображені як зірочки в нижньому шарі, посилають свої сигнали «пристроям виведення даних» (для біологічних нейронів це звичайно м'язи, для штучних - комп'ютерний термінал, за допомогою якого із пристроєм взаємодіє користувач). Посередині ж перебувають нейрони, які напряму не зв'язані із зовнішнім світом (зображені кубами) - вони не одержують із нього інформацію й ніяк на нього не впливають. Вони взаємодіють тільки з іншими нейронами, це і є *приховані шари*.

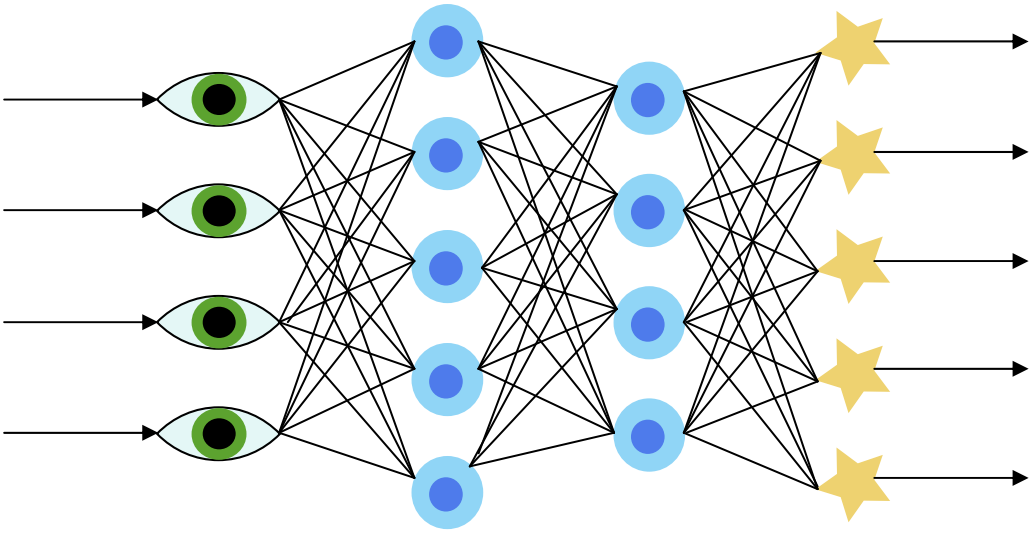


Рисунок 3.1 – Приклад структури нейромережі для розпізнавання образів

У перших штучних нейронних мережах ніяких прихованих шарів не було, і вихідні дані були відносно простою функцією вхідних даних. Такі двошарові «перцептрони» (від лат. perceptio - «сприйняття», математична й комп'ютерна модель сприйняття інформації мозком), що діють винятково на «вхід-вихід», були вкрай обмежені у своїх можливостях. Наприклад, неможливо сконструювати перцептрон, що, зіштовхнувшись із декількома

чорними колами на білому тлі, був би в стані порахувати кількість цих кіл. Лише в 1980-ті, через багато років після перших робіт у цій області, вчені зрозуміли, що включення хоча б одного або двох прихованих шарів разюче розширює можливості штучних нейронних мереж.

В 1981 році Девід Г'юбел і Торстен Візел одержали Нобелівську премію по фізіології й медицині за те, що відкрили механізм дії нейронів у зоровій зоні кори головного мозку. Вони показали, що в прихованих нейронних шарах послідовно витягаються найбільш інформативні властивості візуальних сигналів (наприклад, різкі зміни яскравості або кольорів, що свідчать про границі об'єкта), а потім складають їх у єдине ціле (безпосередньо в об'єкти).

По людських мірках, зір роботів на сьогоднішній день ще залишається досить примітивним. Г'юбел і Візел показали архітектурне рішення, що використовувала Природа, - це архітектура прихованих нейронних шарів. Кожний нейрон у прихованому шарі має матрицю порівняння, що активізується й посилає сигнали в наступний шар, тільки коли інформація, що надходить із попереднього шару, відповідає (з деякою точністю) цій матриці.

У розмові про приховані шари важливо розуміти різницю між простою констатацією ефективності й сили хорошої мережі й складною проблемою того, як таку мережу створити. Одна з найбільших невирішених проблем сучасної науки: як у нейронній мережі з'являються й вкладаються нові приховані шари.

Кількість прихованих шарів та кількість нейронів у кожному прихованому шарі в штучній нейронній мережі на сьогоднішній день підбирається практично випадковим чином, а потім емпіричним шляхом відбувається підбір вдалої архітектури мережі для рішення тієї чи іншої задачі. Як правило кількість нейронів в прихованих шарах беруть більшу, ніж кількість нейронів у вхідному шарі нейронної мережі.

Побудуємо нейронну мережу для розпізнавання літер зображених на рис. 3.2.

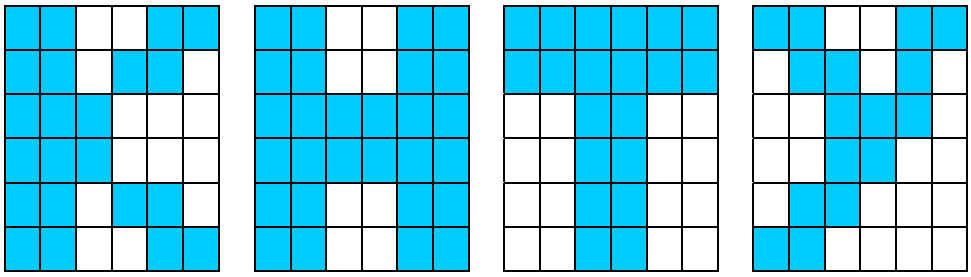


Рисунок 3.2

Кожній літері привласнимо певний код, який нейронна мережа повинна видати як вихідний вектор, при розпізнаванні відповідної літери. Коди літер:

К - 0 0

Н - 0 1

Т - 1 0

У - 1 1

Вхідним вектором нейронної мережі буде значення пікселів чорно-білого зображення літери 6х6, 0 - білий колір, 1 чорний колір. Наприклад, літера Н:

```
1 1 0 0 1 1
1 1 0 0 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 0 0 1 1
1 1 0 0 1 1
```

Для роботи з fannExplorer створимо файли з даними для навчання та тестування.

Файл для навчання test1.train буде містити наступні дані:

4 - чотири образи для навчання

36 - елементів у вхідному векторі (пікселі літери)

2 - елементи у вихідному векторі (дворозрядний код літери)

Зміст файлу test1.train:

```
4 36 2

1 1 0 0 1 1
1 1 0 1 1 0
1 1 1 0 0 0
1 1 1 1 0 0
1 1 0 1 1 0
1 1 0 0 1 1
0 0

1 1 0 0 1 1
1 1 0 0 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 0 0 1 1
1 1 0 0 1 1
0 1

1 1 1 1 1 1
1 1 1 1 1 1
0 0 1 1 0 0
0 0 1 1 0 0
0 0 1 1 0 0
0 0 1 1 0 0
1 0

1 1 0 0 1 1
0 1 1 0 1 0
0 0 1 1 1 0
0 0 1 1 0 0
0 1 1 0 0 0
1 1 0 0 0 0
1 1
```

Файл для тестування test1.test буде містити ті ж самі данні, але дещо викривлені та без вихідних векторів, щоб перевірити здатність мережі до розпізнавання.

Зміст файлу test1.test:

```
4 36 2

0 1 0 0 1 1
1 1 0 1 1 0
1 1 1 0 0 0
1 1 1 1 0 0
1 1 0 1 1 0
1 1 0 0 1 1
0 0

1 0 0 0 1 1
1 1 0 0 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 0 0 1 1
1 0 0 0 1 0
0 0

1 1 1 1 1 1
1 1 1 1 1 1
0 0 1 1 0 0
0 0 1 1 0 0
0 0 1 1 0 0
0 0 1 0 0 0
0 0

1 1 0 0 0 1
0 1 1 0 1 0
0 0 1 1 1 0
0 0 1 1 0 0
0 1 1 0 0 0
1 0 0 0 0 0
0 0
```

Як видно з наведених тестових даних, в перших двох літерах інвертовано декілька пікселів (один в літері К, та два в - Н). Так як формат файлу не дозволяє повністю прибрати вихідні вектори (програма видасть помилку) усі вихідні дані для всіх образів вказані рівними нулю.

Файли test1.train та test1.test необхідно розмістити у наступній папці:

... fann\fannExplorer21\fann\fannSoap\fannKernel\Release\net\

Тепер у програмі fannExplorer створимо нейронну мережу, що буде складатися з трьох шарів:

1 шар - вхідний, 36 нейронів, по одному на піксель зображення.

2 шар - прихований, 36 нейронів.

3 шар - вихідний, два нейрони, для отримання дворозрядного коду літери.

Створити мережу можна через пункт меню File->New Neural Network..., після його натиснення з'явиться діалогове вікно, у якому слід вказати необхідні параметри мережі.

Потім у вкладці Algorithm виберемо параметри нейронів, такі як вид функції активації, алгоритму навчання, тощо. В даному прикладі була вибрана лінійна функція активації та змінено функцію визначення похибки навчання з Tanh на Linear.

Потім слід запустити процес навчання. Після процесу навчання можна перейти до тестування нейронної мережі. Як запускаються процеси навчання та тестування див. л.р. №1.

На рисунку 3 зображене вікно програми fannExplorer після перерахованих вище дій.

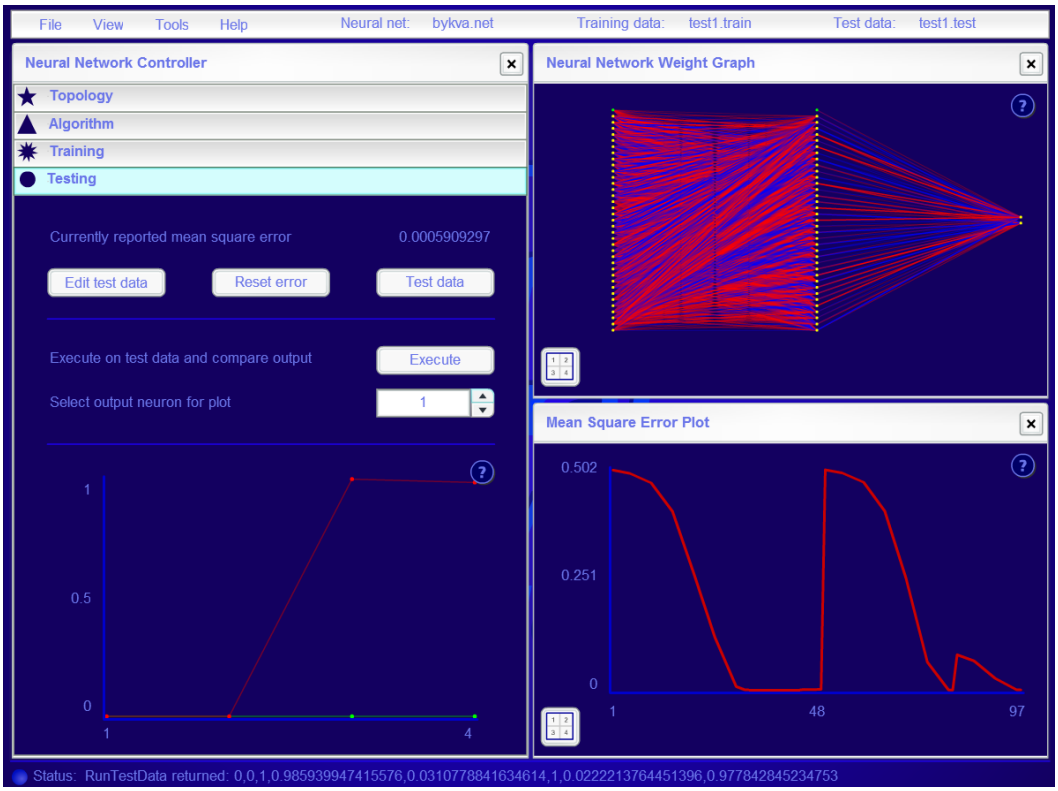


Рисунок 3.3

На вкладці testing можна побачити результати розпізнавання. У полі Select output neuron for plot слід видрати номер вихідного нейрону. Спочатку вибираємо номер 1. На графіку вертикальна вісь - значення виходу нейрону, горизонтальна вісь - номер образу з тестової вибірки.

Зеленим відмічені дані вказані у файлі, вони скрізь рівні нулю так як ми і вказали. Червоним вказані дані одержані в процесі розпізнавання нейронною мережею.

Значення 1-го нейрону (див. рисунок 3.4):

1 образ - вихід першого нейрона 0
2 образ - вихід першого нейрона 0
3 образ - вихід першого нейрона 1
4 образ - вихід першого нейрона 1

Вибираємо другий нейрон у полі Select output neuron for plot.

Значення 2-го нейрону (див. рисунок 3.5):

1 образ - вихід першого нейрона 0
2 образ - вихід першого нейрона 1
3 образ - вихід першого нейрона 0
4 образ - вихід першого нейрона 1

Як бачимо дані вірні, незважаючи на додані помилки.

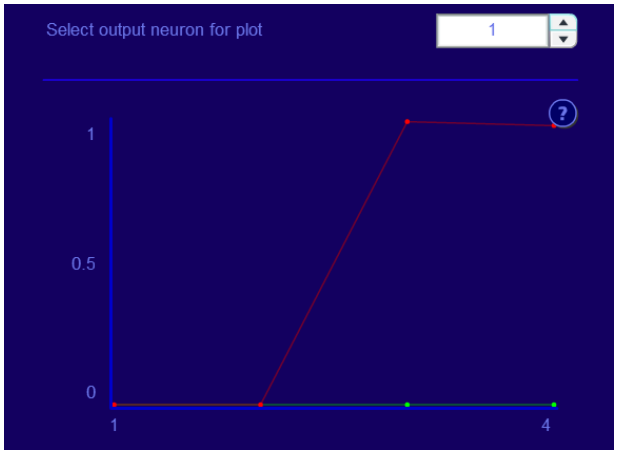


Рисунок 3.4

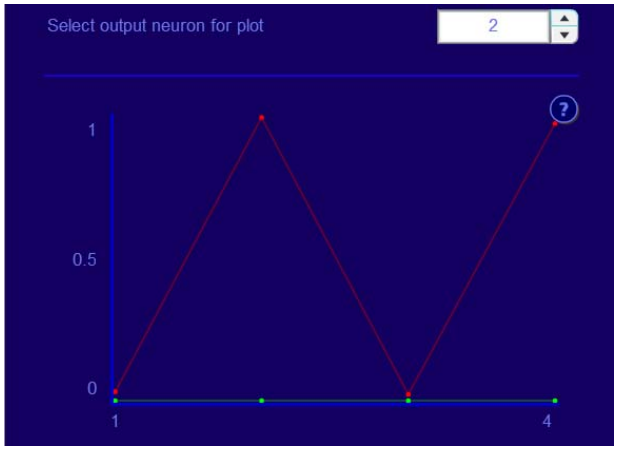


Рисунок 3.5

Завдання:

Побудувати аналогічну нейронну мережу для розпізнавання образів вказаних у Вашому варіанті. Спробувати різну кількість нейронів у прихованому шарі, різну кількість прихованих шарів (без прихованого шару, з декількома прихованими шарами) та різні параметри нейронів (функція активації, алгоритм навчання, функція похибки тощо). Зробити висновки про вплив прихованих шарів та параметрів нейронів на якість розпізнавання образів. Навести хід роботи та скриншоти.

Варіант 1. Образи для розпізнавання - зображення чисел 1, 2, 3.

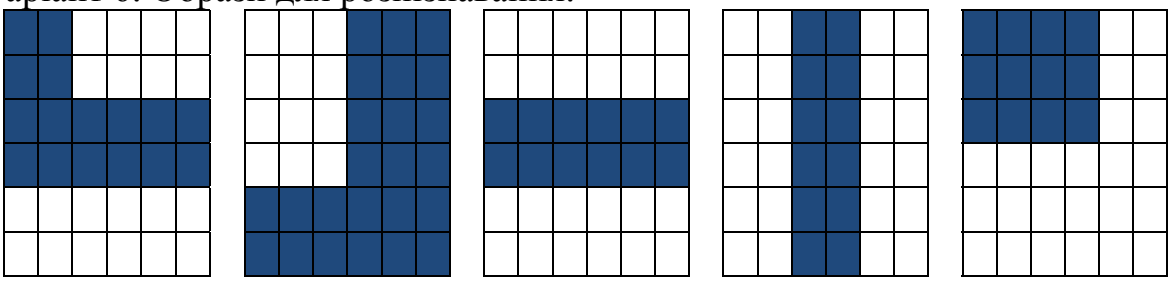
Варіант 2. Образи для розпізнавання - зображення кола, квадрата, ромба, трикутника, еліпса.

Варіант 3. Образи для розпізнавання - зображення літер y, x, z, u.

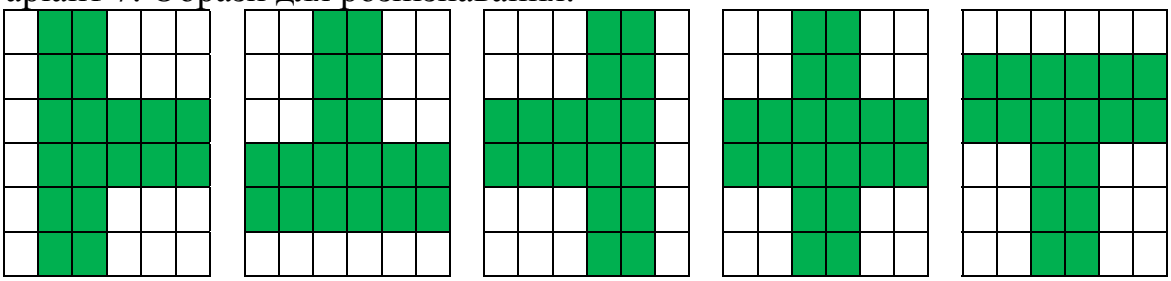
Варіант 4. Образи для розпізнавання - зображення вертикальної лінії, горизонтальної лінії, двох вертикальних ліній, двох горизонтальних ліній.

Варіант 5. Образи для розпізнавання - зображення літер а, б, с, д, е.

Варіант 6. Образи для розпізнавання:



Варіант 7. Образи для розпізнавання:



Контрольні питання:

1. Яку структуру має нейронна мережа для розпізнавання образів?
2. Що таке перцетрон?
3. Що таке шар нейронної мережі?
4. Що таке прихований шар нейронної мережі?
5. На що впливає кількість нейронів та шарів у штучній нейромережі?
6. Як в fannExplorer вводити данні для навчання нейронної мережі?
7. Як в fannExplorer вводити данні для тестування нейронної мережі?

Лабораторна робота №4

Тема: Керування транспортними засобами за допомогою штучних нейронних мереж

Мета: Ознайомитися з методом керування транспортними засобами за допомогою штучних нейронних мереж, розробити архітектуру нейронної мережі для даної задачі та протестувати її у середовищі fannExplorer

Теоретичні відомості

Штучні нейронні мережі можна використовувати для того, щоб певний транспортний засіб (ТЗ) керував собою сам, уникаючи різні перешкоди на своєму шляху. Цього можна домогтися шляхом вибору відповідних входів/виходів і ретельного навчання нейронної мережі. На входи нейронної мережі треба подавати відстані до найближчих перешкод навколо автомобіля, імітуючи зір водія-людини. На виходах треба отримувати прискорення й поворот керма транспортного засобу. Як показує практика – результат може бути вражаючим навіть із використанням усього лише декількох нейронів.

Розглянемо конкретний приклад одної з найпростіших нейронних мереж для керування транспортним засобом (рисунок 4.1). Дана нейронна мережа складається з 4-х прошарків:

– Вхідний прошарок - 3 нейрони.

– 2 приховані прошарки - по 8 нейронів.

– Вихідний прошарок - 2 нейрони.

Зменшення числа прихованих прошарків та нейронів у них призводить до погіршення результату роботи нейронної мережі, а збільшення – до сповільнення її роботи.

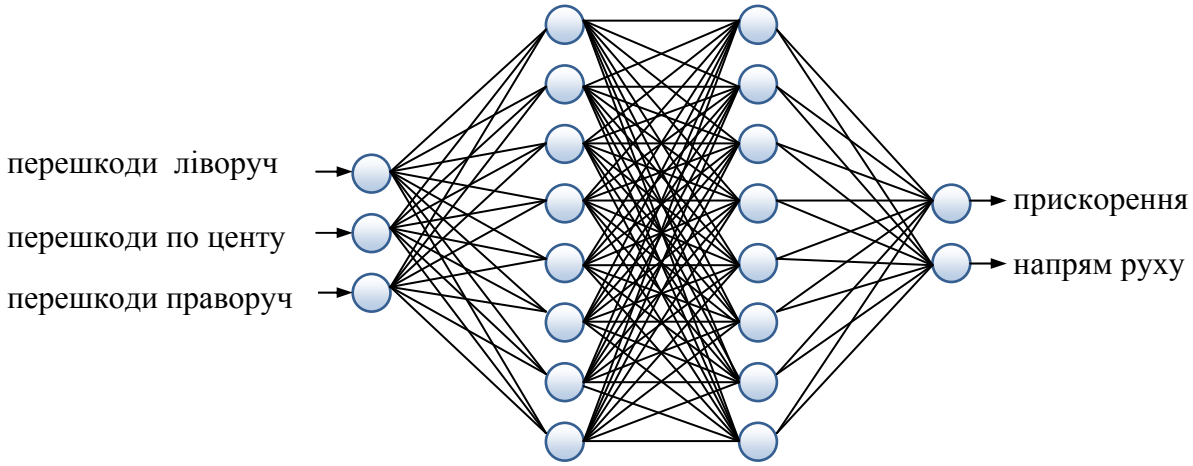


Рисунок 4.1 - Нейронна мережа для керування транспортним засобом

Дана нейронна мережа оперує дробовими числами в діапазоні від 0,0 до 1,0. Розглянемо варіанти вхідних та вихідних даних.

Входи нейронної мережі для керування ТЗ:

1 нейрон - наявність перешкод ліворуч від транспортного засобу.

2 нейрон - наявність перешкод по центру від транспортного засобу.

3 нейрон - наявність перешкод праворуч від транспортного засобу.

Значення входів нейронної мережі для керування ТЗ (наявність перешкоди):

0,0: транспортний засіб торкається перешкоди.

0,1-0,9: транспортний засіб знаходиться на певній відстані від перешкоди, чим більше число – тим більша відстань між ними.

1,0: в полі видимості водія транспортного засобу перешкоди немає.

Виходи нейронної мережі для керування ТЗ:

1 нейрон - значення швидкості транспортного засобу.

2 нейрон - значення напряму транспортного засобу.

Значення виходів нейронної мережі для керування ТЗ:

Прискорення:

0,0-0,2: зворотний хід, чим менше число, тим швидше рух назад.

0,3: швидке гальмування.

0,4: повільне гальмування.

0,5: швидкість без змін.

0,6-0,9: прискорення руху, чим більше число, тим швидше рух вперед.

1,0: повний газ.

Напрям руху:

0,0: повний поворот вліво.

0,1-0,4: частковий поворот вліво, чим більше число – тим менший поворот вліво.

0,5: прямо.

0,6-0,9: частковий поворот вправо, чим більше число – тим більший поворот вправо.

1,0: повний поворот вправо.

Приклад роботи даної нейронної мережі, що пройшла навчання, зображений на рисунку 2.

При вхідному векторі (1.0, 1.0, 0.5) одержано вихідний вектор (0.6, 0.3), тобто при виявленні перешкоди на деякому віддалені праворуч, нейронна мережа подає сигнал транспортному засобу повернути трохи ліворуч та дещо прискорити рух.

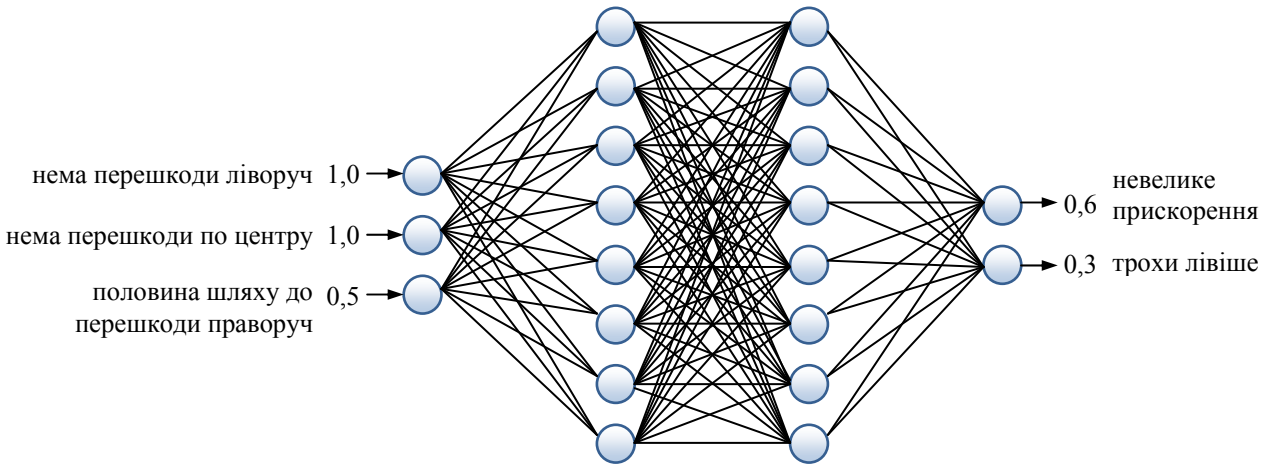


Рисунок 4.2 – Приклад правильної роботи нейронної мережі для керування транспортним засобом

В таблицях 4.1 та 4.2 наведено один з можливих варіантів навчаючої вибірки для даної нейронної мережі. Правила реагування на перешкоди можна вибрати й інші, в залежності від особливостей зовнішнього середовища, в якому буде перебувати ТЗ, та бажаного стилю водіння.

Таблиця 4.1 – Навчаюча вибірка (словесний опис)

Вхідні нейрони			**Вихідні нейрони**	
Відносні відстані до перешкод			*Команди транспортному засобу*	
Ліворуч	*По центру*	*Праворуч*	*Прискорення*	*Напрям руху*
немає перешкод	немає перешкод	немає перешкод	повний газ	прямо
Перешкода далеко	немає перешкод	немає перешкод	невелике прискорення	трохи правіше
немає перешкод	перешкода далеко	немає перешкод	незмінна швидкість	трохи лівіше
немає перешкод	немає перешкод	перешкода далеко	невелике прискорення	трохи лівіше
перешкода далеко	перешкода далеко	немає перешкод	повільне гальмування	трохи правіше
немає перешкод	перешкода далеко	перешкода далеко	повільне гальмування	трохи лівіше
перешкода далеко	немає перешкод	перешкода далеко	прискорення	прямо
перешкода далеко	перешкода далеко	перешкода далеко	швидкість без змін	вліво
перешкода близько	немає перешкод	немає перешкод	швидкість без змін	прямо
немає перешкод	немає перешкод	перешкода близько	швидкість без змін	прямо
немає	перешкода	немає	швидке	повний поворот

Вхідні нейрони			**Вихідні нейрони**	
Відносні відстані до перешкод			*Команди транспортному засобу*	
Ліворуч	*По центру*	*Праворуч*	*Прискорення*	*Напрям руху*
перешкод	близько	перешкод	гальмування	вліво
немає перешкод	перешкода близько	перешкода близько	швидке гальмування	повний поворот вліво
перешкода близько	перешкода близько	немає перешкод	швидке гальмування	повний поворот вправо
перешкода близько	перешкода близько	перешкода близько	зворотний хід	вліво
торкання перешкоди	немає перешкод	немає перешкод	гальмування	повний поворот вправо
немає перешкод	немає перешкод	торкання перешкоди	гальмування	повний поворот вліво
немає перешкод	торкання перешкоди	немає перешкод	зворотний хід	повний поворот вліво
торкання перешкоди	немає перешкод	торкання перешкоди	зворотний хід	повний поворот вліво
торкання перешкоди	торкання перешкоди	немає перешкод	зворотний хід	повний поворот вправо
немає перешкод	торкання перешкоди	торкання перешкоди	зворотний хід	повний поворот вліво
торкання перешкоди	торкання перешкоди	торкання перешкоди	зворотний хід	повний поворот вліво

Таблиця 4.2 – Навчаюча вибірка (адаптована для нейронної мережі)

Вхідні нейрони			**Вихідні нейрони**	
Відносні відстані до перешкод			*Команди транспортному засобу*	
Ліворуч	*По центру*	*Праворуч*	*Прискорення*	*Напрям руху*
1,0	1,0	1,0	1,0	0,5
0,8	1,0	1,0	0,6	0,6
1,0	0,8	1,0	0,5	0,4
1,0	1,0	0,7	0,7	0,3
0,8	0,7	1,0	0,4	0,2
1,0	0,8	0,6	0,4	0,2
0,8	1,0	0,8	0,8	0,5
0,7	0,8	0,8	0,5	0,1
0,2	1,0	1,0	0,5	0,5
1,0	1,0	0,2	0,5	0,5
1,0	0,3	1,0	0,3	0,0
1,0	0,2	0,3	0,3	0,0
0,2	0,2	1,0	0,3	1,0
0,3	0,2	0,2	0,1	0,1
0,0	1,0	1,0	0,4	1,0

Вхідні нейрони			**Вихідні нейрони**	
Відносні відстані до перешкод			*Команди транспортному засобу*	
Ліворуч	*По центру*	*Праворуч*	*Прискорення*	*Напрям руху*
1,0	1,0	0,0	0,4	0,0
1,0	0,0	1,0	0,2	0,0
0,0	1,0	0,0	0,0	0,0
0,0	0,0	1,0	0,1	1,0
1,0	0,0	0,0	0,1	0,0
0,0	0,0	0,0	0,0	0,0

Завдання:

1) Змоделювати нейронну мережу для керування транспортним засобом, розглянуту в теоретичній частині лабораторної роботи, в середовищі fannExplorer. Створити навчаючу та тестову вибірки, використовуючи дані наведені в теоретичній частині лабораторної роботи (в таблицях 4.1-4.2), або запропонувавши свої правила поведінки транспортного засобу. Здійснити процес навчання та тестування даної нейронної мережі. Спробувати різну кількість прихованих прошарків та нейронів у них. Зробити висновки про вплив кількості прихованих шарів і нейронів на якість та час роботи нейронної мережі. Навести одержані при моделюванні скриншоти.

2) Розробити архітектуру нейронної мережі для керування літальним апаратом. В таблиці 4.3 вказані можливі напрями руху для нього.

Таблиця 4.3 – Напрями руху літального апарату

Ліворуч та вгору	Вгору	Праворуч та вгору
Ліворуч на сталій висоті	Прямо на сталій висоті	Праворуч на сталій висоті
Ліворуч та вниз	Внизу	Праворуч та вниз

Змоделювати нейронну мережу для керування літальним апаратом, в середовищі fannExplorer. Створити навчаючу та тестову вибірки. Здійснити процес навчання та тестування даної нейронної мережі. Спробувати різну кількість прихованих шарів та нейронів у них. Зробити висновки про вплив кількості прихованих шарів і нейронів на якість та час роботи нейронної мережі. Експериментальним шляхом підібрати оптимальну кількість прихованих прошарків та прихованих нейронів. Навести схему нейронної мережі, навчаючу та тестову вибірки та одержані при моделюванні скриншоти.

3) Модифікувати нейронну мережу для керування транспортним засобом (з першого завдання) таким чином, щоб додати їй можливість бачити перешкоди позаду (що важливо при зворотному ході), а також можливість повної зупинки при оточенні перешкодами. Навести нову структуру нейронної мережі та нові значення входів і виходів.

Контрольні питання:

Які особливості структури нейронної мережі для керування транспортним засобом?

Які сигнали будуть надходити на вхід нейронної мережі для керування транспортним засобом? А які будуть передаватися на вихід?

Який вигляд може мати навчаюча вибірка для нейронної мережі для керування транспортним засобом?

Що треба модифікувати в нейронній мережі для керування віртуальним автомобілем, щоб її можна було використовувати для керування віртуальним літаком в комп’ютерній грі?

Лабораторна робота №5

Тема: Оптимізація функції із застосуванням генетичних алгоритмів

Мета: Написати програму оптимізації функції з використанням генетичного алгоритму

Теоретичні відомості

На практиці часом складно, а часом і неможливо, зафіксувати властивості функціональної залежності вихідних параметрів від вхідних величин, ще складніше провести аналітичний опис такої залежності. Ця обставина значно ускладнює застосування класичних методів оптимізації, оскільки більшість із них ґрунтуються на використанні апріорної інформації про характер поводження цільової функції, а задача визначення приналежності функції до того або іншого класу порівняна за складністю з початковою.

У зв'язку із цим виникає необхідність побудови таких методів оптимізації, які були б здатні відшукувати рішення при практично повній відсутності припущень про характер досліджуваної функції при рішенні цілочисельних або комбінаторних оптимізаційних задач. Цим вимогам у значній мірі задовольняють еволюційні обчислення, які являють собою алгоритми пошуку, оптимізації або навчання, заснованих на деяких формалізованих принципах природного еволюційного процесу розвитку живих організмів.

Парадигму **генетичних алгоритмів** запропонував Джон Холланд, що опублікував на початку 60-х років її основні положення. А загальне визнання вона одержала після виходу у світ в 1975 році його класичної праці «Адаптація в природних і штучних системах». Генетичний алгоритм був отриманий у процесі узагальнення й імітації в штучних системах таких властивостей живої природи, як природний відбір, пристосовність до мінливих умов середовища, спадкування нащадками життєво важливих властивостей від батьків і т.д.

Тому що алгоритми в процесі пошуку використовують деяке кодування множини параметрів замість самих параметрів, то він ефективно застосовуються для рішення дискретних і комбінаторних задач оптимізації, визначених як на числових множинах, так і на кінцевих множинах довільної природи. Оскільки для роботи алгоритму в якості інформації про оптимізуєму функцію використовуються лише її значення в розглянутих точках простору пошуку, і не потрібно обчислень ні похідних, ні яких-небудь інших характеристик, то даний алгоритм застосуємий до широкого класу функцій, зокрема, не маючих аналітичного опису.

Використання набору початкових точок дозволяє застосовувати для їхнього формування різні способи, що залежать від специфіки розв'язуваної оптимізаційної задачі, у тому числі можливе задання такого набору безпосередньо людиною. Таким чином, основна відмінність генетичних алгоритмів полягає в представленні будь-якої альтернативи рішення у вигляді бітового рядка фіксованої довжини, маніпуляції з якою здійснюються під час відсутності всякого зв'язку з її смисловою інтерпретацією. Тобто в цьому випадку застосовується єдине універсальне представлення будь-якої задачі.

Ідея генетичного алгоритму

Уявимо собі штучний світ, населений множиною істот (особин), причому кожна істота - це деяке рішення нашої задачі. Будемо вважати особину тим більше пристосованою, чим краще відповідне рішення (чим більше значення цільової функції воно дає). Тоді задача максимізації цільової функції зводиться до пошуку найбільш пристосованої істоти. Звичайно, ми не можемо оселити в наш віртуальний світ всі істоти відразу, тому що їхнє число визначається початковою множиною альтернатив. Замість цього ми будемо розглядати багато поколінь, що змінюють один одного.

Тепер, якщо ми зуміємо ввести в дію природний відбір і генетичне спадкування, то отриманий світ буде підкорятися законам еволюції. Помітимо, що, відповідно до нашого визначення пристосованості, метою цієї штучної еволюції буде саме створення найкращих рішень.

Очевидно, еволюція - нескінченний процес, у ході якого пристосованість особин поступово підвищується. Примусово зупинивши цей процес через досить довгий час після його початку й вибравши найбільш пристосовану особину в поточному поколінні, одержимо якщо не абсолютно точну, то близьку до оптимальної відповідь. В цьому й полягає основна ідея генетичного алгоритму. Перейдемо тепер до точних визначень і опишемо роботу генетичного алгоритму більш детально.

Основний зміст генетичного пошуку

Генетичні алгоритми, будучи однією з парадигм еволюційних обчислень, являють собою алгоритми пошуку, побудовані на принципах, подібних із принципами природного відбору й генетики. Якщо говорити узагальнено, вони поєднують у собі принцип виживання найбільш перспективних особин-рішень і структурований випадково-детермінований обмін інформацією, у якому є присутнім елемент випадковості, що моделює природні процеси спадкування й мутації. Додатковою властивістю цих алгоритмів є невтручання людини в розвиваючийся процес пошуку. Людина може впливати на нього лише опосередковано, задаючи певні параметри.

Чим обумовлена популярність генетичних алгоритмів? Як було вже відзначено, ГА дозволяють знайти більш гарні або «раціональні» рішення NP-повних практичних оптимізаційних задач за менший час, ніж інші методи, звичайно застосовувані в цих випадках. Звичайно, термін «гарні» або «раціональні» не строгий у математичному змісті.

Під «раціональними» розуміються рішення, які задовольняють дослідника. Адже в більшості реальних задач немає необхідності знаходити саме глобальний оптимум. Частіше всього метою пошуку є рішення, що задовольняють певним обмеженням. Наприклад, час випробування устаткування не повинний перевищувати певної заданої величини. У цьому сенсі досить знайти саме «раціональне», тобто розумне рішення. Друга важлива причина росту популярності ГА полягає в стрімкому зростанні продуктивності сучасних комп'ютерів.

Переваги **генетичних алгоритмів** стають більш очевидними, якщо розглянути основні їхні відмінності від традиційних методів. Основних відмінностей п'ять:

1. Генетичні алгоритми працюють із кодами, у яких представлений набір параметрів, що прямо залежать від аргументів цільової функції.
2. Для пошуку генетичний алгоритм використовує декілька точок пошукового простору одночасно (розпаралелення), а не переходить від точки до точки, як це робиться в традиційних методах. Тобто ГА оперують одночасно з усією сукупністю припустимих рішень.
3. Генетичні алгоритми в процесі роботи не використовують ніякої додаткової інформації, що підвищує швидкість його роботи.
4. Генетичний алгоритм використовує як ймовірнісні правила для породження нових точок пошуку, так і детерміновані правила для переходу від одних точок до інших.
5. Генетичні алгоритми здійснюють пошук оптимального рішення за однією й тою ж стратегією, як для унімодальних, так і для багатоекстремальних функції.

Генетичний алгоритм працює з **кодовими послідовностями** (КП) — кодами безвідносно їхньої значеннєвої інтерпретації. Тому сама КП і її структура описуються поняттям **генотип**, а його інтерпретація, з погляду розв'язуваної задачі, поняттям **фенотип**. Кожна КП представляє, по суті, точку простору пошуку. Екземпляр кодової послідовності називають хромосомою, особиною або індивідуумом.

У принципі, ГА не обмежені бінарним або цілочисельним поданням. Відомі й інші реалізації, побудовані винятково на векторах речовинних числах. Незважаючи на те, що для багатьох реальних завдань більше підходять рядки змінної довжини, у даний час структури фіксованої довжини найпоширеніші й вивчені.

На кожному кроці роботи генетичний алгоритм використовує декілька точок пошуку одночасно. Сукупність цих точок є набором кодових послідовностей (особин), які утворюють початкову множину рішень — **К** (популяцію). Кількість особин у популяції називають **розміром популяції**. На кожному кроці роботи генетичний алгоритм обновляє початкову множину **К** шляхом створення нових КП і знищення «безперспективних», не задовольняючих критерію цільової функції. Кожне відновлення інтерпретується як зміна поколінь і звичайно ідентифікується за заданим розміром.

У процесі роботи алгоритму генерація нових особин відбувається на основі моделювання процесу розмноження. При цьому, природно, що породжуючі особини називаються батьками, а породжені — нащадками. Батьківська пара, як правило, породжує пари нащадків. Безпосередня генерація нових рядків із двох обраних відбувається за рахунок роботи **оператора схрещування** (випадково-детермінованого обміну), що у процесі роботи алгоритму може застосовуватися не до всіх пар батьків.

Частина цих пар може переходити в популяцію наступного покоління безпосередньо. Наскільки часто буде виникати така ситуація, залежить від ймовірності застосування оператора схрещування, що є одним з параметрів генетичного алгоритму.

Моделювання процесу генерації нових точок пошуку здійснюється за рахунок роботи **оператора мутації**, що задається певною ймовірністю. Оскільки еволюційний процес біологічних видів супроводжується загибеллю останніх, то породження нащадків повинне супроводжуватися знищенням інших безперспективних особин. Вибір пар батьків з популяції для породження нащадків робить **оператор відбору**, а вибір особин для знищення - **оператор редукції**.

Характеристики генетичного алгоритму вибираються таким чином, щоб забезпечити малий час роботи, з одного боку, і пошук якомога кращого рішення, з іншого.

Загальний вид генетичного алгоритму

Незважаючи на те, що теорія генетичних алгоритмів в інформатиці з'явилася порівняно недавно, цей напрямок було швидко підхоплено вченими. І до теперішнього часу існує досить велика кількість їхніх різновидів еволюційних алгоритмів. Проте їхню основу становить базова модель, представлена на рис. 5.1.

Рисунок 5.1 – Базовий генетичний алгоритм

Різниця полягає лише в генетичних операціях, представлених усередині зазначеного алгоритму. Виключення становлять гібридні алгоритми, у яких до зазначених блоків можуть додаватися елементи класичних алгоритмів оптимізації.

Далі розглянемо детально методологію побудови генетичного алгоритму.

Створення хромосом

Хромосома представляє собою набір зістикованих значень змінних, представлений у двійковому коді. Розглянемо метод кодування:

Нехай X описується діапазоном значень [Xmin...Xmax], з точністю E.

Спочатку обчислюється кількість значень N, яких необхідно закодувати, за формулою:

$$N = \frac{Xmax - Xmin}{E} . \quad (5.1)$$

Потім підбирається таке значення кількості бітів NB, що задовольняє умові:

$$2^{NB} \geq N. \quad (5.2)$$

Далі обчислюється крок дискретизації заданого діапазону:

$$D = \frac{Xmax - Xmin}{NB} . \quad (5.3)$$

Знаючи крок дискретизації ми можемо закодувати значення X натуральним числом Nx за формулою:

$$Nx = \frac{X}{D} . \quad (5.4)$$

Після чого Nx кодується у двійковій системі числення.

У випадку мінімізації функції багатьох змінних двійкові коди кожної змінної зістиковуються в певній послідовності й зберігається інформація про кількість бітів, якою кодується кожна змінна.

Функція пристосованості

Під функцією пристосованості розуміється деяка цільова функція, екстремум якої й потрібно відшукати. Однак, у деяких літературних джерелах вказується, що ця функція є зворотною від цільової функції у випадках коли потрібно відшукати мінімум. Але це вносить певні незручності в перевірці рішення. Тому під функцією пристосованості краще розуміти саме цільову функцію. Якщо доводиться використовувати генетичний алгоритм для навчання нейронних, нейронечітких мереж або ж добірки коефіцієнтів ПІД-регуляторів за зразком (навчальною вибіркою), то як функція пристосованості звичайно виступає класична функція суми квадратів помилки F:

$$F = \sum_i (Y_i - Y_i')^2, \quad (5.5)$$

де i - номер елемента навчальної вибірки;

Y - вектор значень вихідний змінної;

Y' - вектор відповідних значень мережі (регулятора).

Ініціалізація початкової популяції

На цьому етапі визначається кількість хромосом (особин) у популяції K. У різних джерелах літератури вказується, що воно залежить від кількості вхідних змінних і звичайно вибирається за правилом:

$$K \geq 2M, \tag{5.6}$$

де M - кількість вхідних змінних.

Далі задаються початкові значення хромосомам. Якщо відомі деякі наближення до мінімумів, то деяким хромосомам привласнюються ці значення. Всім іншим значення задаються випадково. Потім обчислюються відповідні їм значення функції пристосованості.

Селекція

Суть операції селекції полягає у відборі пар хромосом для рекомбінації. І може здійснюватися трьома способами:

1. Ранжирування.
2. Випадковий вибір.
3. Виважено випадковий вибір.

При ранжируванні всі хромосоми впорядковуються за убуванням їхньої функції пристосованості й пари розбиваються послідовно за принципом «краща із кращої».

При випадковому виборі пари утворюються випадковим чином.

При зважено випадковому виборі кожній хромосомі привласнюється певний бал залежний від функції пристосованості. Потім всі хромосоми розташовуються послідовно на шкалі суми балів з діапазоном свого бального значення. Пари формуються шляхом випадкового вибору числа на шкалі. Отже чим більше бал у хромосоми, тим більше в неї шансів. Даний спосіб іноді ще називають «принципом рулетки».

Рекомбінація (кросовер)

На даному етапі відбувається обмін генетичною інформацією. Рекомбінація є найбільш важливим генетичним оператором. Вона генерує нові хромосоми, поєднуючи генетичний матеріал двох батьківських. Існує декілька варіантів рекомбінації. Найбільш простим є одноточковий. У цьому варіанті просто беруться дві хромосоми, і розрізаються у випадково обраній точці. Результуючі хромосоми виходять із початку однієї й кінця іншої батьківських хромосом.

001100101110010|11000	-->	0011001011100101***11100***
110101101101000|11100		***110101101101000***11000

Мутація

Мутація являє собою випадкову зміну хромосоми (звичайно простою зміною стану одного з бітів на протилежний). Даний оператор дозволяє більш швидко знаходити локальні екстремуми з одного боку, і дозволяє "перескочити" на інший локальний екстремум з іншого.

00110010111001***0***11000	-->	00110010111001***1***11000

У деяких генетичних алгоритмах до мутації може додаватися операція інверсії.

Інверсія інвертує (змінює) порядок бітів у хромосомі шляхом циклічної перестановки (випадкова кількість разів).

00110010111001***0******11000***	-->	***11000***00110010111001***0***

Умова зупинки

Умова зупинки залежить від того, де ми застосовуємо генетичний алгоритм. Залежно від цього можна виділити наступні умови:

- Цільова функція досягла деякої заданої точності або значення. Цей випадок підходить для алгоритмів навчання із учителем нейронних або нейронечітких мереж, настроювання коефіцієнтів ПІД-регуляторів або інших математичних апаратів, пов'язаних з навчанням за зразком (навчальній вибірці).
- Досягнуто задане число поколінь. Цей випадок застосовується в тих випадках, коли система явно обмежена часом і обчислювальними ресурсами на пошук оптимального рішення. Звичайно це потрібно в завданнях адаптації систем у динамічному режимі.
- Протягом заданого числа поколінь оптимальне значення цільової функції не змінюється. Такий варіант можливий у випадках, коли алгоритм досяг якого-небудь глобального або сильно вираженого локального екстремума. Це умова необхідно, коли алгоритм використовується як для навчання із учителем (вихід із зациклення у випадку неможливості відшукати рішення із заданою точністю), так і при пошуку екстремума, значення неможливо визначити заздалегідь.

Умови, при виконанні яких задача вирішується ефективно генетичними алгоритмами:

- великий простір пошуку, ландшафт якого є негладким (містить трохи екстремумів);
- складність формалізації оцінки якості рішення функцією ступеня придатності;
- багатокритеріальність пошуку;
- пошук прийнятного рішення за заданими критеріями на відміну від пошуку єдиного оптимального.

Основні завдання, які можуть ефективно вирішувати генетичні алгоритми, можна звести до наступних класів:

- навчання нейронних і нейронечітких мереж із учителем;
- адаптація нейронних і нейронечітких мереж у динамічному режимі;
- настроювання коефіцієнтів ПІД-регуляторів;
- рішення класичних оптимізаційних завдань (завдання лінійного програмування, транспортні завдання й т.д.);
- рішення комбінаторних завдань.

Завдання:

Вибрати функцію й діапазон відповідно до варіанта. Побудувати графік функції. Написати програму знаходження максимуму й мінімуму функції на заданому діапазоні. Проаналізувати отримані результати.

Варіанти:

№	Функція
1.	Y(x)=x*sin(5*x), x=[-2...5]
2.	Y(x)=1/x*sin(5*x),x=[-5...5]
3.	Y(x)=2^x*sin(10x),x=[-3...3]
4.	Y(x)=x^(1/2)*sin(10*x),x=[0...5]
5.	Y(x)=15*sin(10*x)*cos(3*x),x=[-3...3]
6.	Y(x)=5*sin(10*x)*sin(3*x),x=[0...4]
7.	Y(x)=sin(10*x)*sin(3*x)/(x^2),x=[0...4]
8.	Y(x)=5*sin(10*x)*sin(3*x)/(x^(1/2)),x=[1...7]
9.	Y(x)=5*cos(10*x)*sin(3*x)/(x^(1/2)),x=[0...5]
10.	Y(x)=-5*cos(10*x)*sin(3*x)/(x^(1/2))x=[0...10]
11.	Y(x)=-5*cos(10*x)*sin(3*x)/(x^x),x=[0...5]
12.	Y(x)=5*sin(10*x)*sin(3*x)/(x^x),x=[0...8]
13.	Y(x)=x^sin(10*x),x=[1...10]
14.	Y(x)=-x^cos(5*x),x=[0...10]
15.	Y(x)=x^cos(x^2),x=[0...10]
16.	Y(x)=cos(x^2)/x,x=[0...5]
17.	Y(x)=10*cos(x^2)/x^2,x=[0...4]
18.	Y(x)=(1/x)*cos(x^2+1/x),x=[1...10]
19.	Y(x)=sin(x)*(1/x)*cos(x^2+1/x),x=[-2...2]
20.	Y(x)=5*sin(x)*cos(x^2+1/x)^2,x=[1...10]
21.	Y(x)=5*sin(1/x)*cos(x^2+1/x)^2,x=[1...4]
22.	Y(x)=5*sin(1/x)*cos(x^2)^3,x=[-4...4]
23.	Y(x)=(x^3)*cos(x^2),x=[-2...2]
24.	Y(x)=(x^3)+cos(15*x),x=[-2...2]
25.	Y(x)=(3^x)+cos(15*x),x=[-1...2]

Контрольні питання:

1. Що таке генетичні алгоритми? Для чого їх можна застосовувати?
2. З яких етапів складається генетичний алгоритм?
3. Яким чином кодується особина в генетичному алгоритмі?
4. Що таке функція пристосованості?
5. Як реалізується селекція?
6. Як реалізується кросовер?
7. Як реалізується мутація?
8. Якою може бути умова зупинки роботи генетичного алгоритму?
9. При виконанні яких умов задача вирішується ефективно генетичними алгоритмами?

Лабораторна робота №6

Тема: Інтелектуальні агенти. Алгоритм Q-навчання

Мета: Ознайомитися з поняттям інтелектуального агента та одним з методів його навчання – Q-learning

Теоретичні відомості

Інтелектуальний агент (у штучному інтелекті) – сутність, яка одержує інформацію через систему сенсорів про стан зовнішнього середовища (керованих нею процесів) та здійснює вплив на нього, при цьому її реакції раціональні в тому розумінні, що її дії сприяють досягненню певної мети. Найбільш близьким аналогом у живій природі є примітивне інстинктивне поводження комах. Термін «інтелектуальний» не означає наявності інтелекту, але підкреслює більш високий рівень технології керування в порівнянні із примітивними тригерними системами автоматичного керування. Такий агент може бути як програмною системою, так і складною автоматизованою системою, наприклад, верстатом з ЧПУ або комплексом керування технологічними, логістичними, фінансовими або будь-якими іншими процесами. Про "інтелектуальність" агента можна говорити, якщо його взаємодія з навколишнім середовищем є адекватною тій або іншій системі вимог. Ніякого відношення навіть до інтелекту вищих тварин і вже тим більше людини подібна функціональність не має.

У штучному інтелекті існує декілька типів агентів. Наприклад:

1) фізичний агент – агент, що сприймає навколишній світ через деякі сенсори й діє за допомогою маніпуляторів.

2) часовий агент – агент, що використовує інформацію, яка змінюється з ходом часу, і пропонує деякі дії або надає дані комп'ютерній програмі чи людині.

Бувають різні типи інтелектуальних агентів. Найпростіші з них діють по схемі:

IF (умова) THEN дія;

їх дії жорстко прописані.

Більш складні агенти можуть мати інформацію про зовнішнє середовище та принципи його функціонування, знати ступінь корисності тої чи іншої своєї дії, навчатися на основі інформації, одержуваної із зовнішнього середовища.

Агенти, що навчаються

У деякій літературі агенти, що навчаються (АН) *називаються автономними інтелектуальними* агентами (англ. *autonomous intelligent agents*), що підкреслює їхню незалежність і здатність до навчання й пристосовування до мінливих обставин.

Алгоритм Q-навчання інтелектуальних агентів

Q-навчання (Q-learning) **-** метод навчання інтелектуального агента, заснований на системі підкріплень (винагород від зовнішнього середовища при виборі вірних дій).

Спочатку агент випадковим чином вибирає свою поведінку, після чого отримує реакцію від середовища, на основі якої формує функцію корисності Q своїх дій. Функція Q дає йому можливість надалі уже не випадково вибирати стратегію поведінки, а враховувати досвід попередньої взаємодії із середовищем.

Одна з переваг Q-навчання – воно дає можливість порівнювати очікувану корисність доступних дій, не формуючи моделі навколишнього середовища.

Даний алгоритм оперує поняттями **дія** й **стан** інтелектуального агента.

У розпорядженні агента є деяка множина **дій** $A(a_1, a_2 \dots a_n)$. Дії агента впливають на середовище, і агент може визначати, у якому стані він перебуває в поточний момент, і одержувати винагороду від середовища за правильні дії.

Зовнішнє середовище представлене множиною можливих **станів** $S(s_1, s_2 \dots s_n)$, в яких може перебувати агент.

Цільовий стан агента - це такий стан, досягнення якого є кінцевою метою діяльності агента.

Завданням агента є знайти найкращу стратегію для досягнення **цільового стану**. В даному алгоритмі вона описується Q-значеннями, які визначають корисність виконуваної дії у відповідному стані.

Суть алгоритму Q-learning

Матриця R – містить дані про зовнішнє середовище. Матриця R представляє собою матрицю суміжності графа, в якому вершини – це можливі стани агента, а ребра – дії агента, які переводять його з одного стану в інший.

Матриця Q – пам’ять агента, що містить інформацію про корисність його дій. Це двомірна матриця, в якій кожний стан агента співставлений з певною величиною винагороди, яку одержить агент за перехід у даний стан.

При ініціалізації програми матриця Q заповнюються нулями, якщо агент не має жодних знань про навколишнє середовище, або даними з матриці R, якщо початкові знання у нього є. Не маючи інформації про винагороди від того чи іншого стану агент обирає першу дію випадковим чином.

Формула оновлення матриці Q після кожної дії агента:

$$\mathbf{Q[s, a] = R[s, a] + Gamma * MAX(Q[s', a']),}$$

де Q[s, a] – комірка матриці Q, що відповідає поточному стану агента;

R[s, a] – комірка матриці R, що відповідає поточному стану агента;

Gamma – швидкість навчання, рекомендоване значення = 0.8;

Q[s', a'] – комірка матриці Q, що відповідає наступному стану агента;

MAX(Q[s', a']) – вибір з множини можливих дій агента в поточному стані дії з максимальною винагородою.

Приклад 1. Дано схему будинку (рисунок 6.1). Інтелектуальний агент - робот, що повинен навчитися виходити на вулицю з будь-якої кімнати і бажано найкоротшою відстанню. Отримати знання про розташування кімнат та дверей він може лише на практиці, блукаючи по будинку.

Припустимо, що у нас є будинок з 5 кімнатами, які з'єднані дверима, як показано на рисунку 1. Пронумеруємо кожну кімнату від 0 до 4, а вулиці привласнимо номер 5.

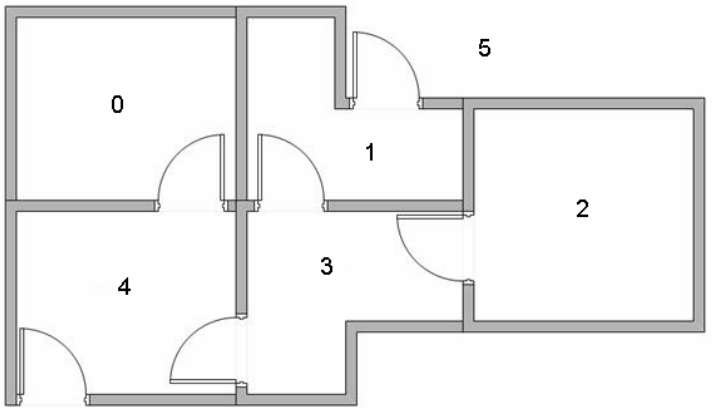


Рисунок 6.1

В даному прикладі станом агента буде перебування його в тій чи іншій кімнаті, або вулиці, а дією - вихід/вхід у ті чи інші двері.

Тобто в агента є 6 можливих станів:

0 стан - перебування у кімнаті №0;

1 стан - перебування у кімнаті №1;

2 стан - перебування у кімнаті №2;

3 стан - перебування у кімнаті №3;

4 стан - перебування у кімнаті №4;

5 стан - перебування на вулиці.

Цільовий стан - 5 стан (перебування на вулиці).

Представимо зовнішнє середовище агента у вигляді графа, в якому стани - вузли, а дії - ребра (рисунок 6.2).

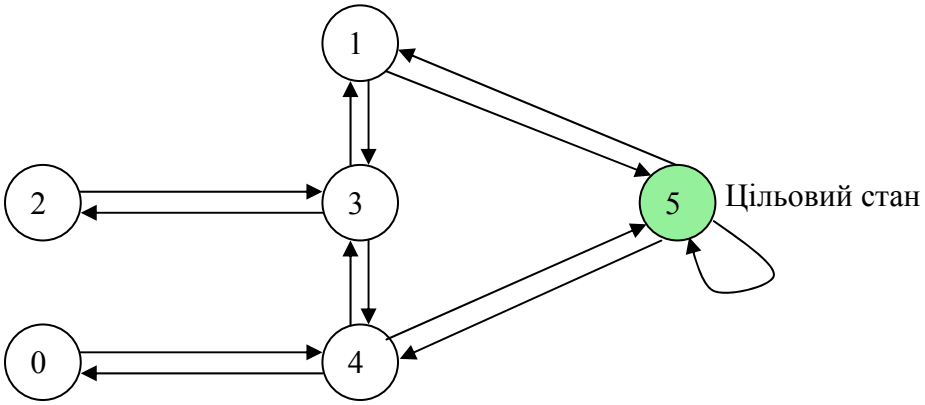


Рисунок 6.2

В якій би кімнаті не опинився агент, він повинен знайти вихід за межі будівлі. А якщо агент вже знаходиться на вулиці, то повинен залишатися там - тому 5-та вершина має петлю (рисунок 6.2).

Привласнимо кожному ребру вагу. Ребра, що ведуть до цільового стану, мають одержати вагу 100, всі інші - вагу 0 (рисунок 6.3).

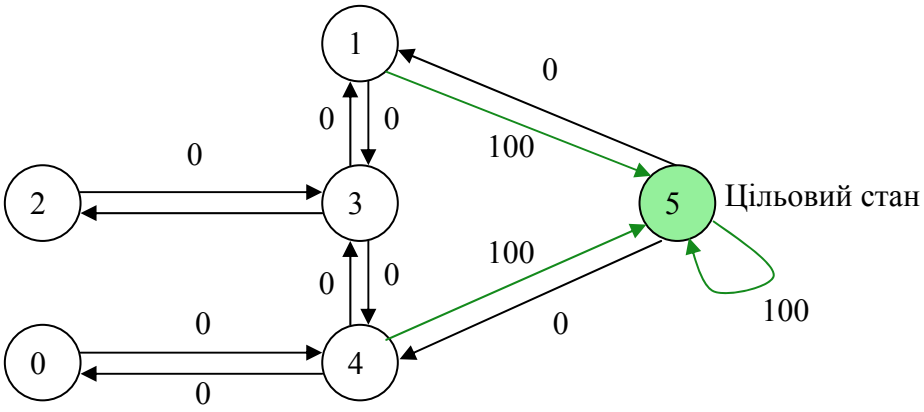


Рисунок 6.3

Тепер побудуємо матрицю суміжності даного графа (рисунок 6.4). Якщо вершини не зв’язані ребром, ставимо біля них у матриці (-1). Одержана матриця і є R матрицею.

		Стан					
		0	1	2	3	4	**5**
Дія	0	-1	-1	-1	-1	0	-1
	1	-1	-1	-1	0	-1	100
	2	-1	-1	-1	0	-1	-1
	3	-1	0	0	-1	0	-1
	4	0	-1	-1	0	-1	100
	5	-1	0	-1	-1	0	100

Рисунок 6.4

Індекси рядків R матриці вказують на номер стану агента, а індекси стовпчиків - на номер дії.

Розглянемо покроково одну з можливих послідовностей дій агента.

Ініціалізація.

1) Ініціалізація матриці R:

R=

	0	1	2	3	4	**5**
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	0	-1
4	0	-1	-1	0	-1	100
5	-1	0	-1	-1	0	100

Рисунок 6.5

2) Ініціалізація матриці Q (для наочності припустимо, що пам’ять агента вже містить записи про розташування цільового стану):

Q =

	0	1	2	3	4	**5**
0	0	0	0	0	0	0
1	0	0	0	0	0	100
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	100
5	0	0	0	0	0	100

Рисунок 6.6

3) Ініціалізація параметра швидкості навчання Gamma = 0,8.

1 крок. Агент випадковим чином опинився у кімнаті №3.

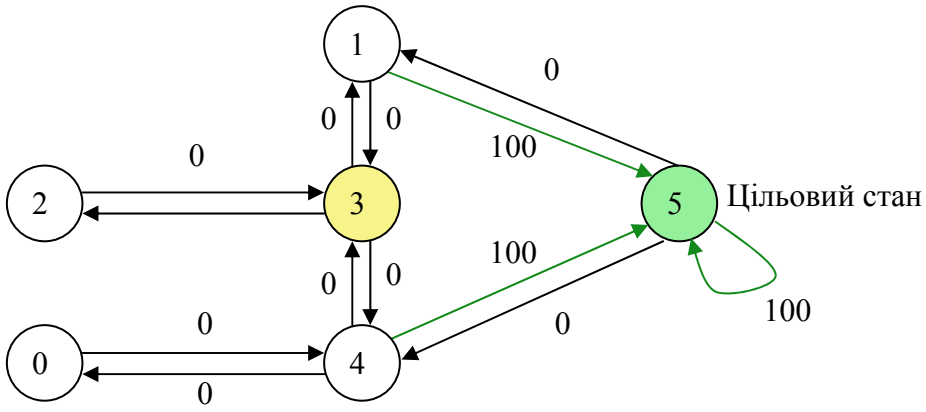


Рисунок 6.7

З матриці R агент дізнається про свої можливі дії. Оскільки агент перебуває в 2 стані, то його можливі дії містяться в 3-му рядку матриці. Зі стану 3 агент може перейти в стани: 1, 2 або 4, тобто здійснити дії (3, 1), (3, 2) або (3, 4). Оскільки агент ще не знає, який стан наблизить його до цільового стану, він обирає дію випадково. Припустимо, що агент випадковим чином обрав дію (3, 1).

R=

	0	1	2	3	4	**5**
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	0	-1	100
2	-1	-1	-1	0	-1	-1
3	-1	**0**	**0**	-1	**0**	-1
4	0	-1	-1	0	-1	100
5	-1	0	-1	-1	0	100

Рисунок 6.8

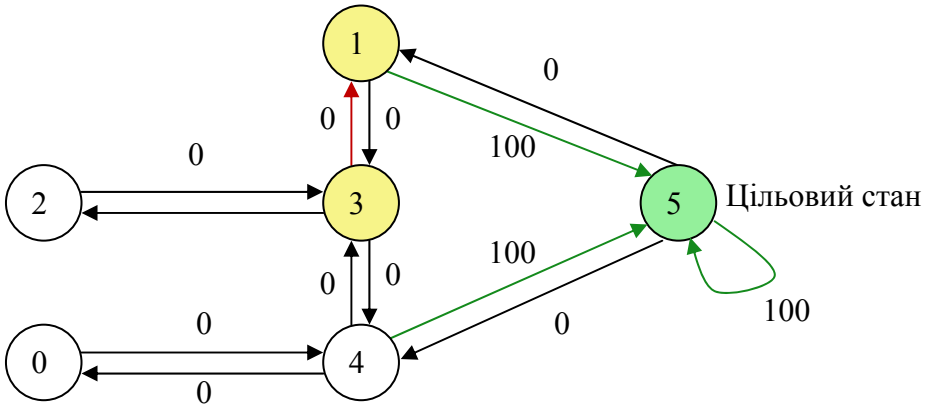


Рисунок 6.9

2 крок. На другому кроці агент обирає наступну дію та одержує винагороду за попередню дію.

Зараз агент перебуває в 1 стані. З даного стану він може перейти в стан 3 та стан 5. Оскільки стан п’ять відмічений вагою 100, то агент розуміє, що даний стан є цільовим. Отже агент видирає перехід у цільовий стан, тобто дію (1, 5).

R=

	0	1	2	3	4	**5**
0	-1	-1	-1	-1	0	-1
1	-1	-1	-1	**0**	-1	**100**
2	-1	-1	-1	0	-1	-1
3	-1	0	0	-1	0	-1
4	0	-1	-1	0	-1	100
5	-1	0	-1	-1	0	100

Рисунок 6.10

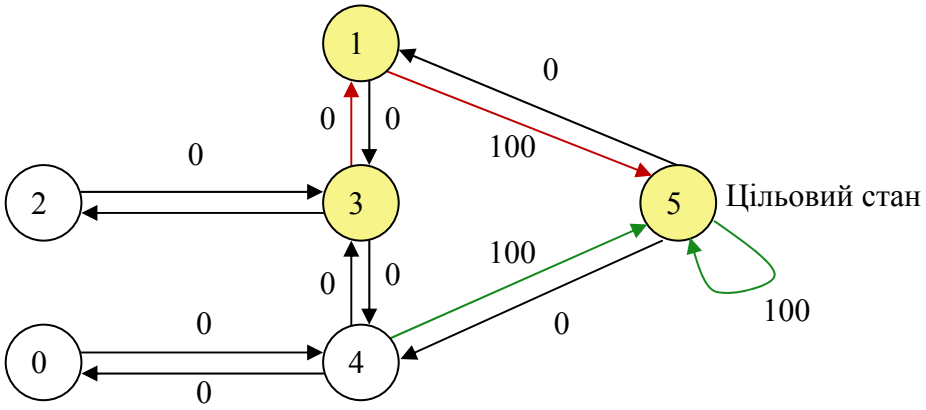


Рисунок 6.11

Винагорода за попередню дію (здійснену на кроці 1) обчислюється по наступній формулі:

Q[3, 1] = R[3, 1] + 0,8 * MAX(Q[1, 3], Q [1, 5]) = 0 + 0,8 * 100 = 80

В матрицю Q, тобто пам'ять агента записується інформація про корисність дії (3, 1).

Q =

	0	1	2	3	4	**5**
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	**80**	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

Рисунок 6.12

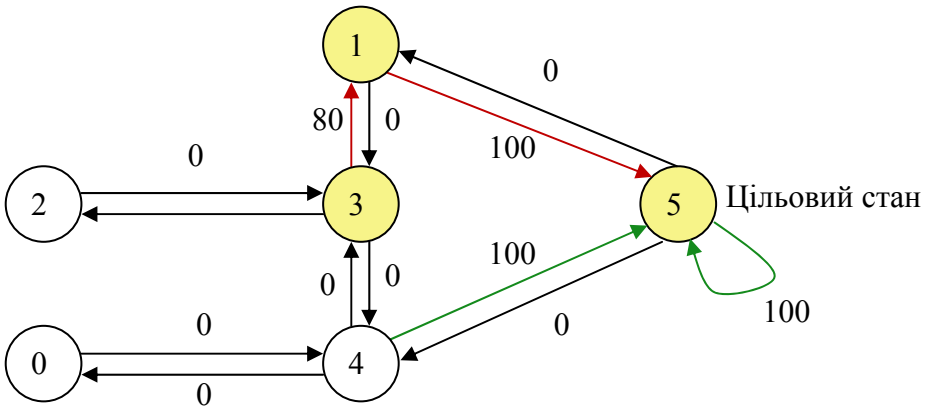


Рисунок 6.13

3 крок. На другому кроці агент знову обирає наступну дію та одержує винагороду за попередню дію.

Агент перебуває в 5 стані. Можливі дії - (5, 5), (5, 1) , (5, 4). Агент обирає дію (5, 5) як цільову дію.

Винагорода за попередню дію (здійснену на кроці 2) обчислюється по наступній формулі:

Q[1, 5] = R[1, 5] + 0,8 * MAX(Q[5, 5], Q [5, 1], Q [5, 4]) = 100 + 0,8 * 100 = 180

Q =

	0	1	2	3	4	**5**
0	0	0	0	0	0	0
1	0	0	0	0	0	**180**
2	0	0	0	0	0	0
3	0	80	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

Рисунок 6.14

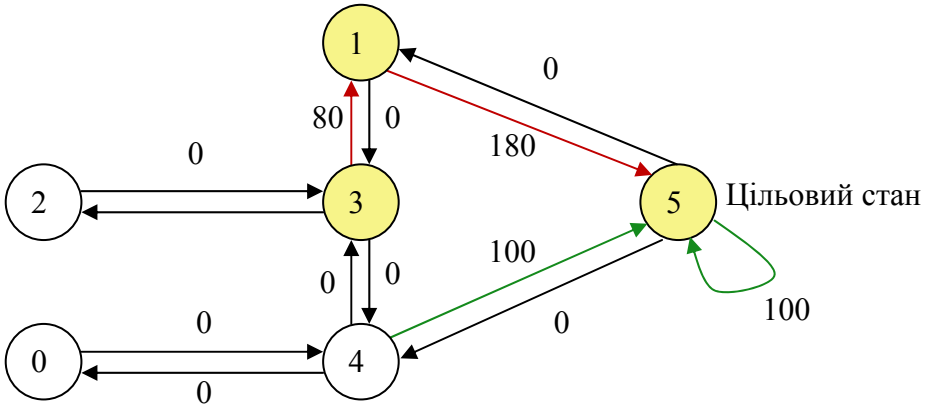


Рисунок 6.15

4 крок. Обчислимо винагороду за перебування в 5 стані та завершимо виконання алгоритму.

Q[5, 5] = R[5, 5] + 0,8 * MAX(Q[5, 5], Q [5, 1], Q [5, 4]) = 100 + 0,8 * 100 = 180

Q =

	0	1	2	3	4	**5**
0	0	0	0	0	0	0
1	0	0	0	0	0	180
2	0	0	0	0	0	0
3	0	80	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	**180**

Рисунок 6.16

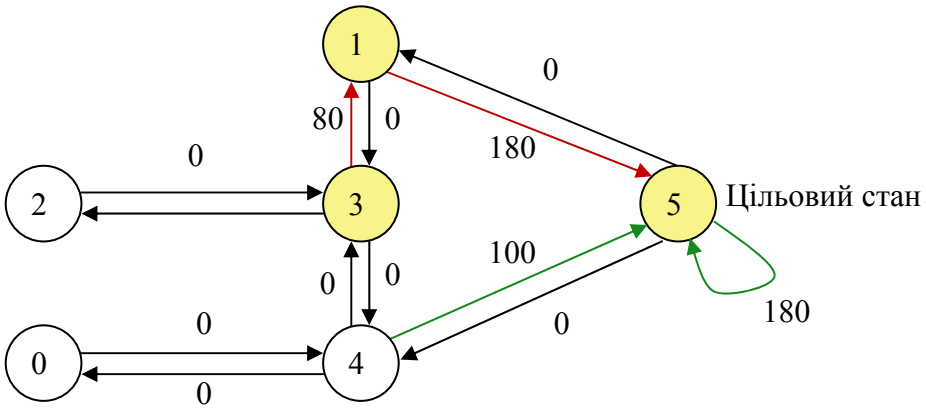


Рисунок 6.17

Тепер у пам’яті агента міститься інформація про корисність дій (3, 1), (1, 5) та (5, 5).

Чим більше разів агент буде шукати з різних кімнат вихід на вулицю, тим більше інформації запишеться в його пам’ять.

Наприклад, якщо агент здійснить 6 успішних спроб вийти з будинку з різних кімнат:

1 спроба -1 стан, 5 стан;

2 спроба - 3 стан, 4 стан, 5 стан;

3 спроба - 5 стан, 5 стан;

4 спроба - 2 стан, 0 стан, 4 стан, 5 стан;

5 спроба - 4 стан, 5 стан;

6 спроба - 0 стан, 4 стан, 5 стан.

То його пам’ять виглядатиме наступним чином:

Q =

	0	1	2	3	4	**5**
0	0	0	0	0	144	0
1	0	0	0	247	0	386
2	0	0	0	0	0	0
3	0	221	0	0	309	0
4	115	0	0	115	0	386
5	0	309	0	0	309	386

Рисунок 6.18

Завдання 1

Відповідно до свого варіанта побудувати математичну модель інтелектуального агента та його зовнішнього середовища:

– записати множину станів інтелектуального агента та цільовий стан;

– зобразити зовнішнє середовище агента у вигляді графа, в якому стани – вузли, а дії – ребра;

– записати матрицю суміжності побудованого графа.

Проілюструвати роботу алгоритму Q-навчання розрахунками без програмної реалізації. Для цього показати 2 спроби агента досягти цільової мети. Перший раз, обираючи маршрут цілком довільним чином (так як пам’ять агента порожня), другий раз – враховуючи здобуту пам’ять агента. Розрахунки повинні містити обчислення винагород за дії агента, значення матриці Q та зображення графа зовнішнього середовища з позначенням шляху агента і змінених значень ваг ребер.

Завдання 2

Відповідно до свого варіанта та побудованої в 1 завданні математичної моделі реалізувати програмне забезпечення для ілюстрації роботи алгоритму Q-навчання. Примітка: на початку програми ініціалізувати матрицю Q нулями.

Додаток А6. Варіанти завдання

Варіант 1

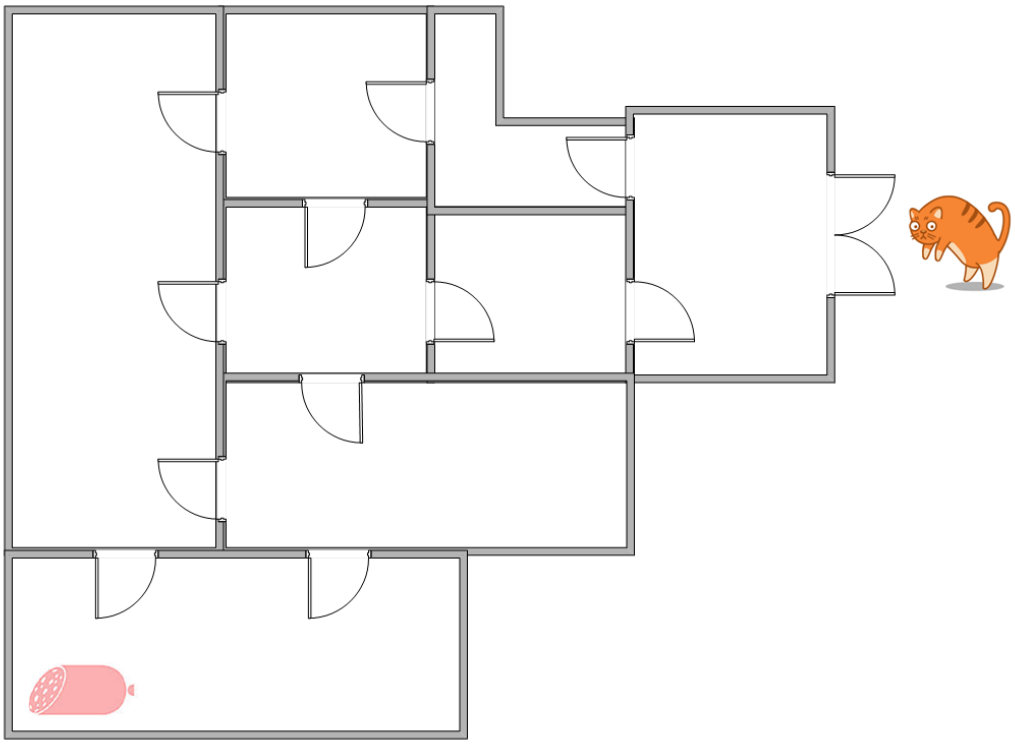


Рисунок А6.1

Варіант 2

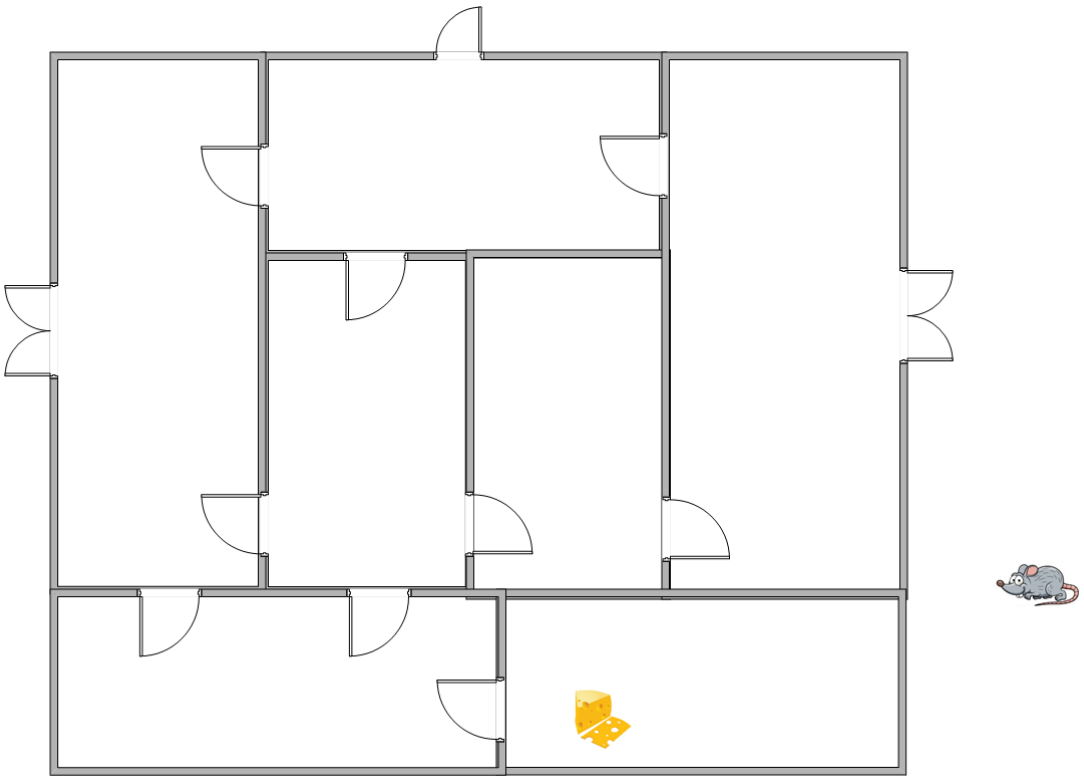


Рисунок А6.2

Варіант 3

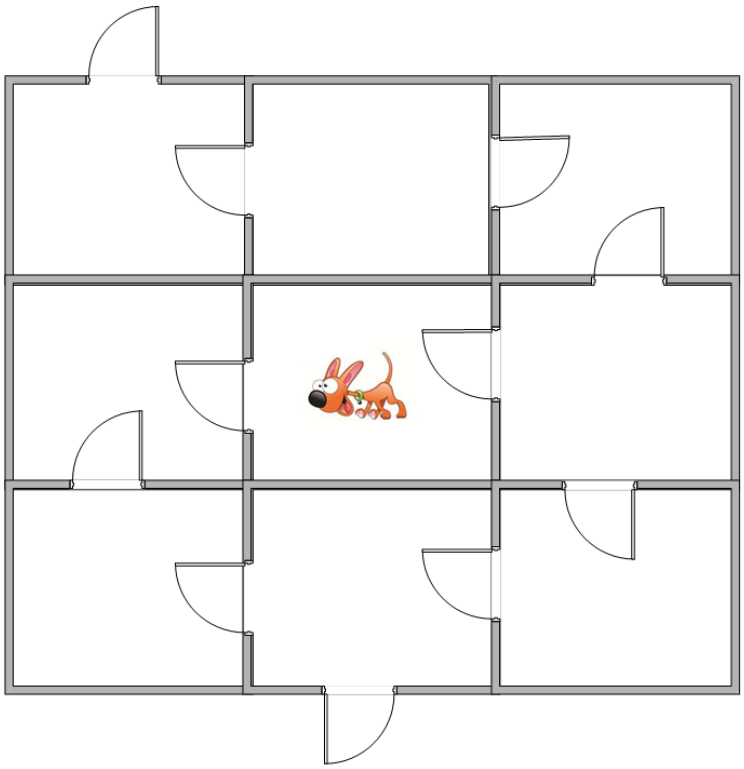


Рисунок А6.3

Ціль інтелектуального агента - вийти на вулицю.

Варіант 4

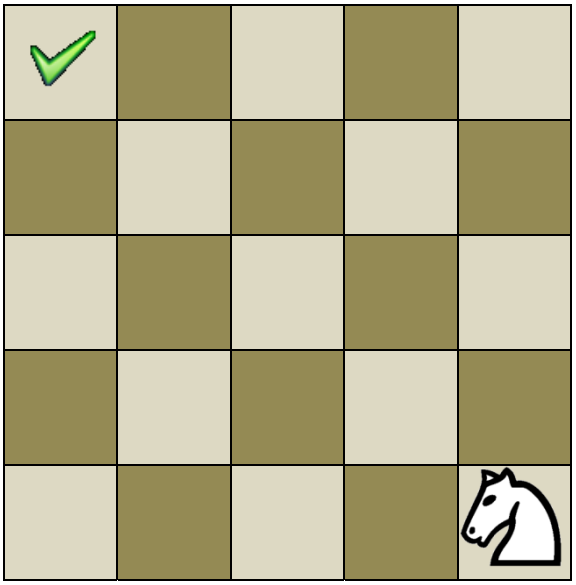


Рисунок А6.4

Агент може пересуватися по клітинках за правилами гри в шахи (агент - фігура кінь).

Варіант 5

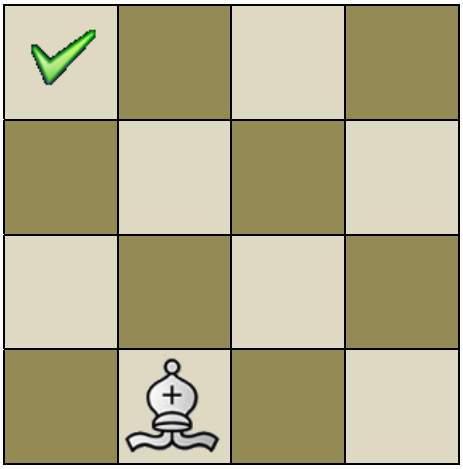
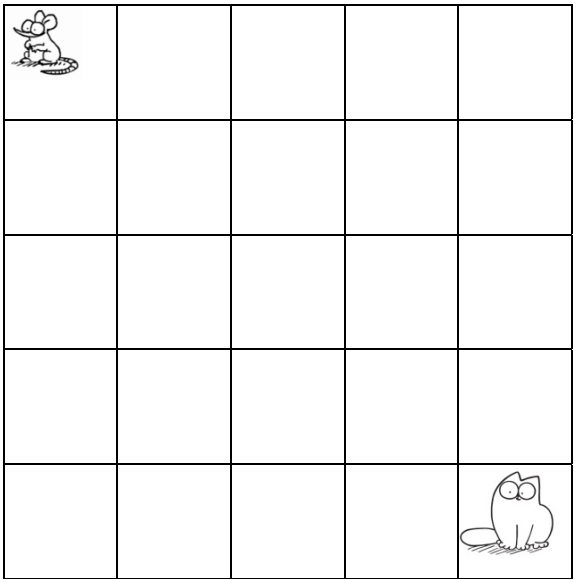


Рисунок А6.5

Агент може пересуватися по клітинках за правилами гри в шахи (агент - фігура слон).

Варіант 6



Агент може пересуватися по клітинках наступним чином:

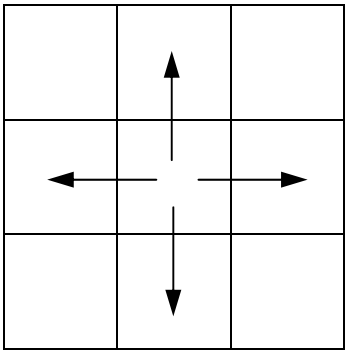
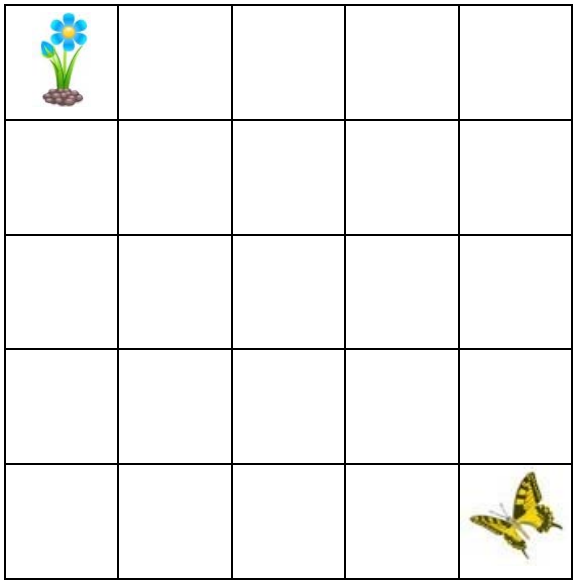


Рисунок А6.6

Варіант 7



Агент може пересуватися по клітинках наступним чином:

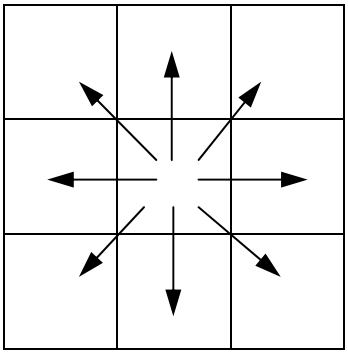
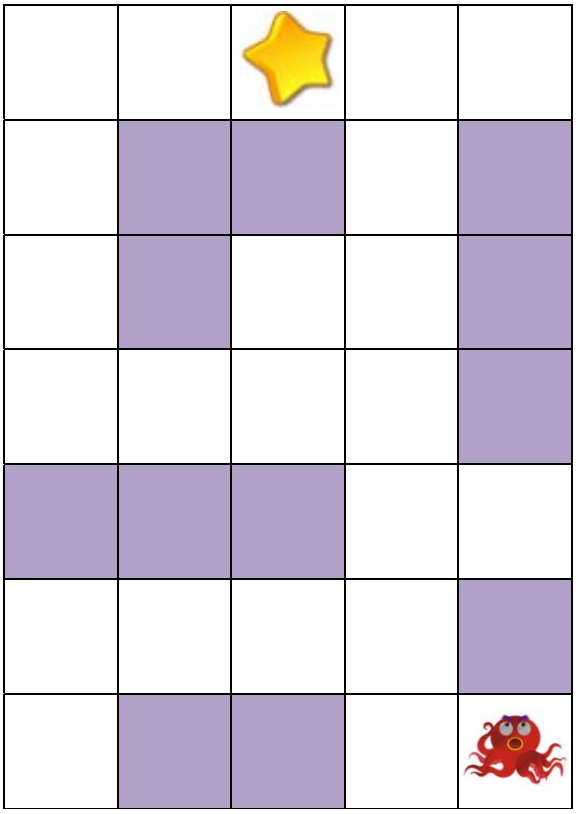


Рисунок А6.7

Варіант 8



Клітинки фіолетового кольору - перешкоди.

Агент може пересуватися по клітинках наступним чином:

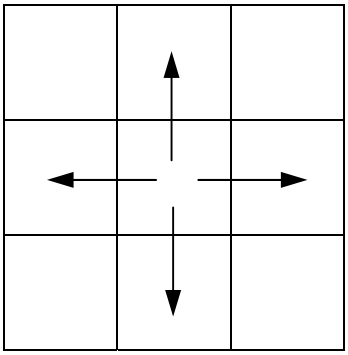
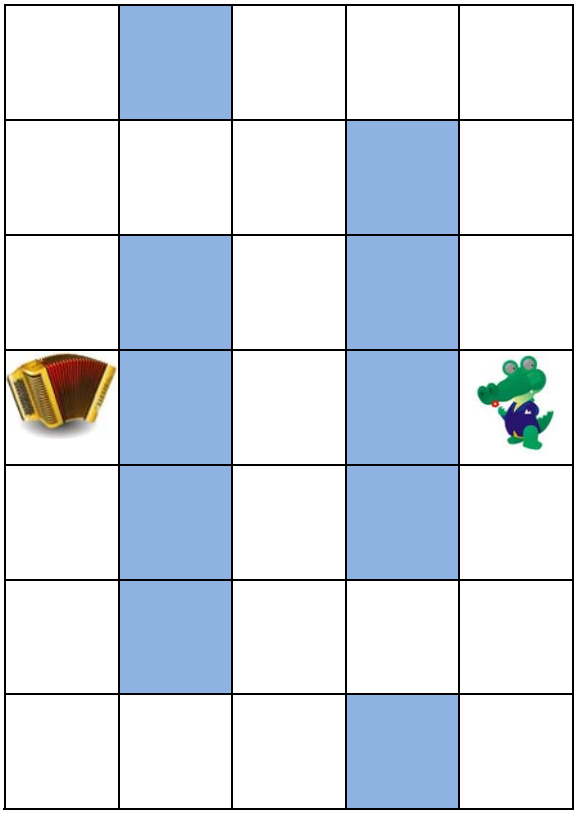


Рисунок А6.8

Варіант 9



Клітинки синього кольору - перешкоди.

Агент може пересуватися по клітинках наступним чином:

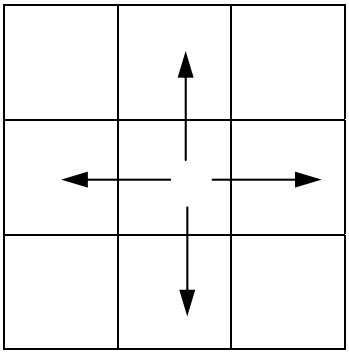
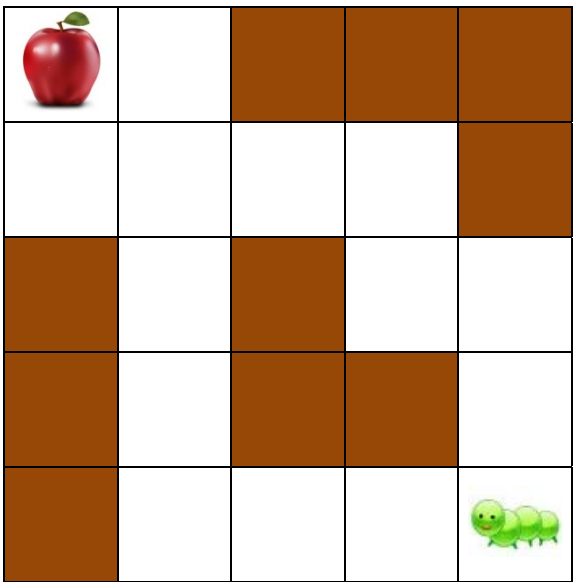


Рисунок А6.9

Варіант 10



Клітинки коричневого кольору - перешкоди.

Агент може пересуватися по клітинках наступним чином:

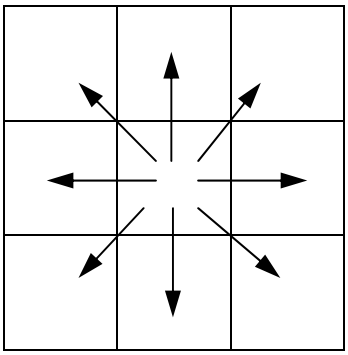


Рисунок А6.5

Контрольні питання:

1. Що таке інтелектуальний агент?
2. Які бувають інтелектуальні агенти?
3. У чому полягає алгоритм Q-навчання?
4. Для чого застосовується матриця R в алгоритмі Q-навчання?
5. Для чого застосовується матриця Q в алгоритмі Q-навчання?
6. Яким чином представляється зовнішнє середовище інтелектуального агента в алгоритмі Q-навчання?

Лабораторна робота №7

Тема: Клітинний автомат «Гра життя»

Мета: Написати програму, що реалізує клітинний автомат «Гра життя».

Теоретичні відомості

Клітинні автомати є дискретними динамічними системами, поведінка яких повністю визначається в термінах локальних залежностей, в значній мірі те саме відноситься і до великого класу безперервних динамічних систем, визначених рівняннями в часткових похідних. У цьому сенсі клітинні автомати в інформатиці є аналогом фізичного поняття «поле».

Клітинний автомат може мислитися як стилізований світ. Простір представлено рівномірною сіткою, кожна комірка якої містить декілька бітів даних, час йде вперед дискретними кроками, а закони світу виражаються єдиним набором правил, скажімо, невеликою довідковою таблицею, за якою будь-яка клітина на кожному кроці обчислює свій новий стан за станам її близьких сусідів. Таким чином, закони системи є локальними і всюди однаковими.

Якщо заданий відповідний набір правил, то такий простий операційний механізм достатній для підтримання цілої ієрархії структур і явищ. Клітинні автомати дають корисні моделі для багатьох досліджень в природничих і обчислювальних науках і комбінаторній математиці, вони, зокрема, представляють природний шлях вивчення еволюції великих фізичних систем. Клітинні автомати до того ж утворюють загальну парадигму паралельних обчислень, подібно тому, як це роблять машини Тюрінга для послідовних обчислень.

Клітинні автомати винаходилися багато разів під різними назвами, і трохи різні один від одного поняття вживалися під однією і тією ж назвою. У чистій математиці їх можна виявити як один з розділів топологічної динаміки, в електротехніці вони іноді називаються ітеративними масивами, а в інформатиці це вид гри на домашньому комп'ютері.

Клітинні автомати, мабуть, знайшли стійке (і все більш важливе) застосування в якості концептуальних і практичних моделей просторово розподілених динамічних систем, для яких фізичні системи є першими і найбільш важливими прототипами.

Основний напрям дослідження клітинних автоматів – алгоритмічна вирішуваність певних задач. Також розглядаються питання побудови початкових станів, при яких клітинний автомат вирішуватиме задану задачу.

«Гра життя» – клітинний автомат, вигаданий англійським математиком Джоном Конвейем (John Horton Conway) 1970.

Місце дії цієї гри – «всесвіт» – це площина, поділена на клітинки. Кожна клітинка на цій поверхні може знаходитись в двох станах: бути живою або бути мертвою. Клітинка має вісім сусідів. Розподіл живих клітинок на початку гри називається першим поколінням. Кожне наступне покоління утворюється на основі попереднього за наведеними нижче правилами.

Правила

– якщо у живої клітини два чи три сусіди – то вона лишається жити;

– якщо у живої клітини один чи немає сусідів – то вона помирає від «самотності»;

– якщо у живої клітини чотири та більше сусідів – вона помирає від «перенаселення»;

– якщо у мертвої клітини рівно три сусіди – то вона оживає.

Дані правила отримали назву генетичних законів Конвея, вони задовольняють трьом основним умовам:

1) не має бути жодної початкової конфігурації, для якої існувало б просте доведення можливості необмеженого росту популяції;

2) мають існувати такі початкові конфігурації, які заздалегідь володіють властивістю безмежно розвиватися;

3) мають існувати прості початкові конфігурації, які протягом значного проміжку часу ростуть, перетерплюють різноманітні зміни та закінчують свою еволюцію одним з трьох наступних способів:

a. повністю щезають;

b. переходять у стійку конфігурацію та перестають змінюватися взагалі;

c. виходять у коливальний режим з певним періодом.

Гравець не бере прямої участі у грі, а лише розставляє початкову конфігурацію «живих» клітин, які потім взаємодіють відповідно до правил уже без його участі.

Фігури

Ці прості правила призводять до виникнення величезної кількості різноманітних форм, кожна з яких має дещо спільне з попередньою. На цей час склалася така система їхньої класифікації:

Стійкі фігури – фігури, які залишаються незмінними за кожної ітерації

Періодичні фігури – фігури, стан яких повторюється через деяку кількість поколінь.

Фігури, що рухаються – фігури у яких стан повторюється, але з деяким зсувом у просторі.

Гармати – фігури у яких стан повторюється, але кожен цикл вони додатково створюють фігури, що рухаються.

Паротяги – фігури, що рухаються, які залишають за собою сліди у вигляді стійких або періодичних фігур.

Пожирачі – стійкі фігури, які при зіткненні з деякими фігурами, що рухаються, зберігають свій стан, знищуючи рухому фігуру.

У цій грі "швидкістю світла" називають швидкість шахового короля. Очевидно, що з такими правилами жодна взаємодія не може передаватися з більшою швидкістю.

Приклади

Незабаром після публікації правил, було виявлено декілька цікавих фігур, зокрема: *r*-пентаміно, глайдер (англ. *glider*).

Нерухомі фігури

Нерухомі фігури не змінюються з плином часу. Найпростіший приклад нерухомої фігури – блок.

Рисунок 7.1 – Нерухома фігура блок

Осцилятори

Осцилятор – така фігура, що має певну періодичність. Приклад: лінія з 3-х клітин.

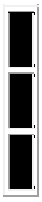


Рисунок 7.2 – Періодична фігура смужка

Планери

Планери (англ. *glider*) – рухомі фігури, які є періодичними, але з кожним циклом руху вони зміщуються на декілька клітин у певному (зазвичай сталому) напрямку.

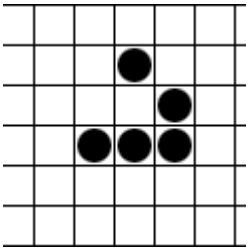


Рисунок 7.3 – Класичний планер

Фігура планер в 2003 році була запропонована в якості емблеми хакерів.

Гармата планерів

Гармата планерів – періодична фігура, яка за повний цикл генерує один чи декілька планерів.

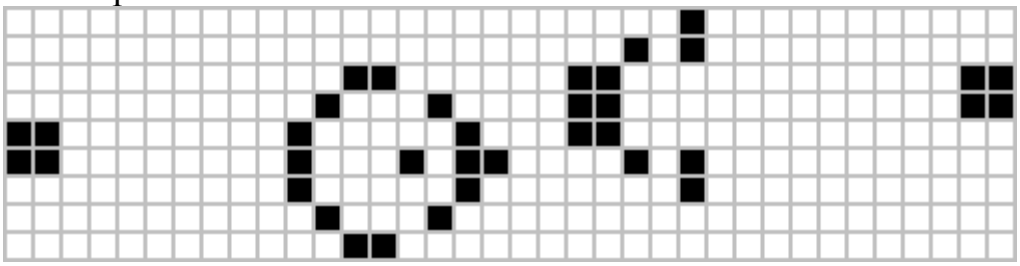


Рисунок 7.4 – Гармата планерів (глайдерна гармата)

Конвей спочатку припустив, що ніяка початкова комбінація не може привести до необмеженого розмноження і запропонував премію в 50 доларів тому, хто доведе або спростує цю гіпотезу. Приз був отриманий групою з

Массачусетського технологічного інституту, яка придумала нерухому повторювану фігуру, яка періодично створювала рухомі «планери». Таким чином, кількість живих клітин могла рости необмежено. Потім були знайдені рухомі фігури, що залишають за собою «сміття» з інших фігур.

Едемський сад

Едемським садом називається таке розташування клітин, у якого не може бути попереднього покоління. Практично для будь-якої гри, стан кліток, в якій визначається декількома сусідами на попередньому кроці, можна довести існування садів Едему, хоча побудова конкретної фігури є набагато складнішою.

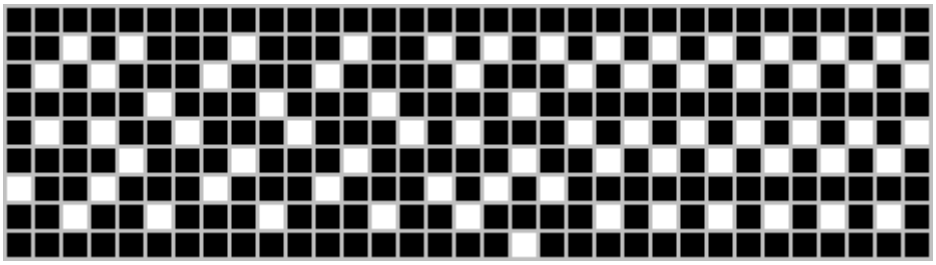


Рисунок 7.5 – Приклад Едемського саду

Завдання:

Реалізувати клітинний автомат «Гра життя». Використати для тестування програми фігури наведені в додатку А.

Додаток А7

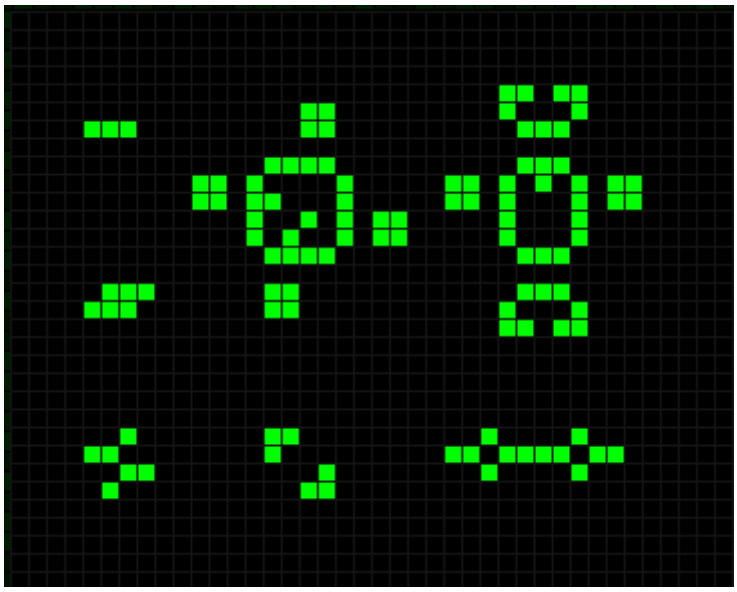


Рисунок А7.1 – Приклад осцелярів

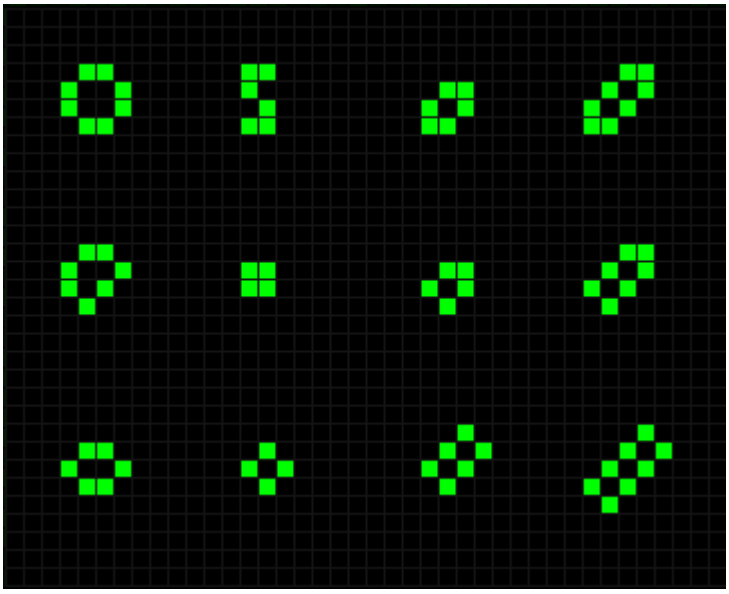


Рисунок А7.2 – Приклад стійких фігур

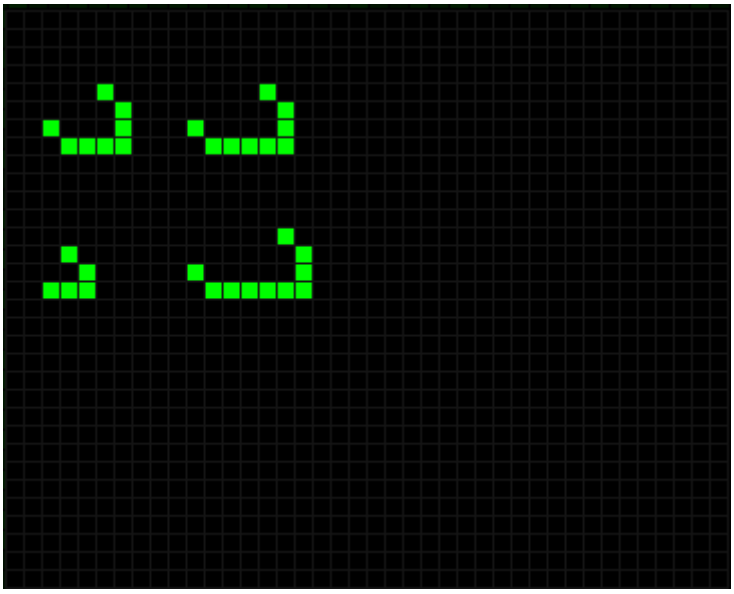


Рисунок А7.3 – «Космічні кораблі»

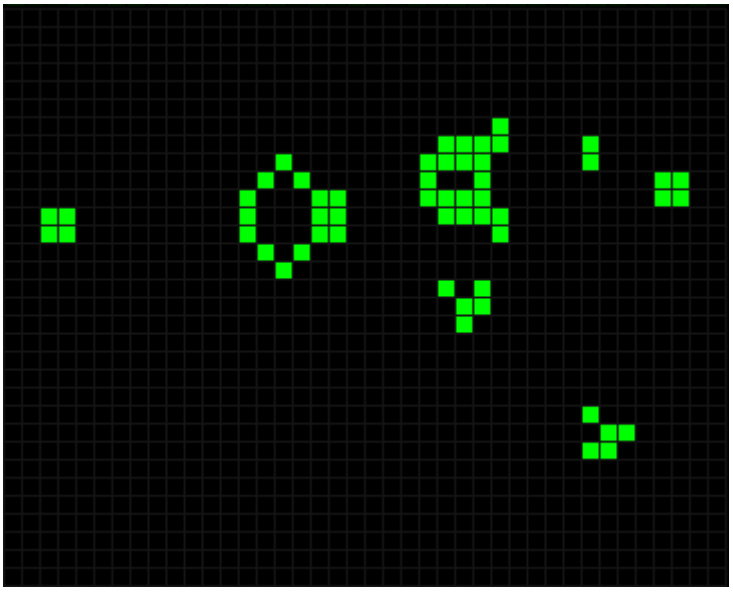


Рисунок А7.4 – «Планерна гармата»

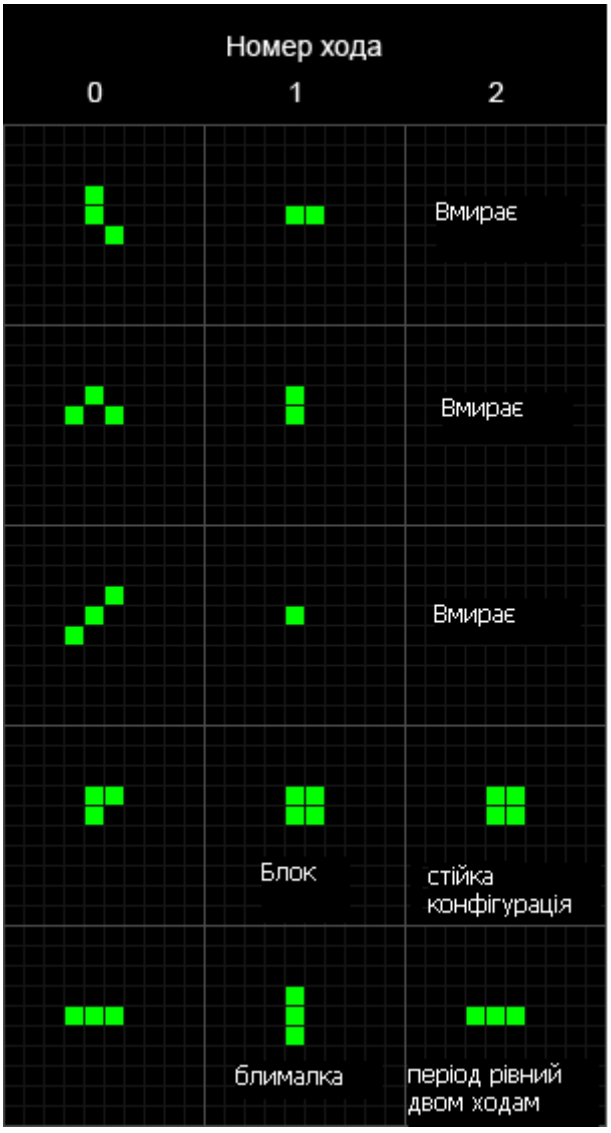


Рисунок А7.5 – Приклади комбінацій з трьох фішок

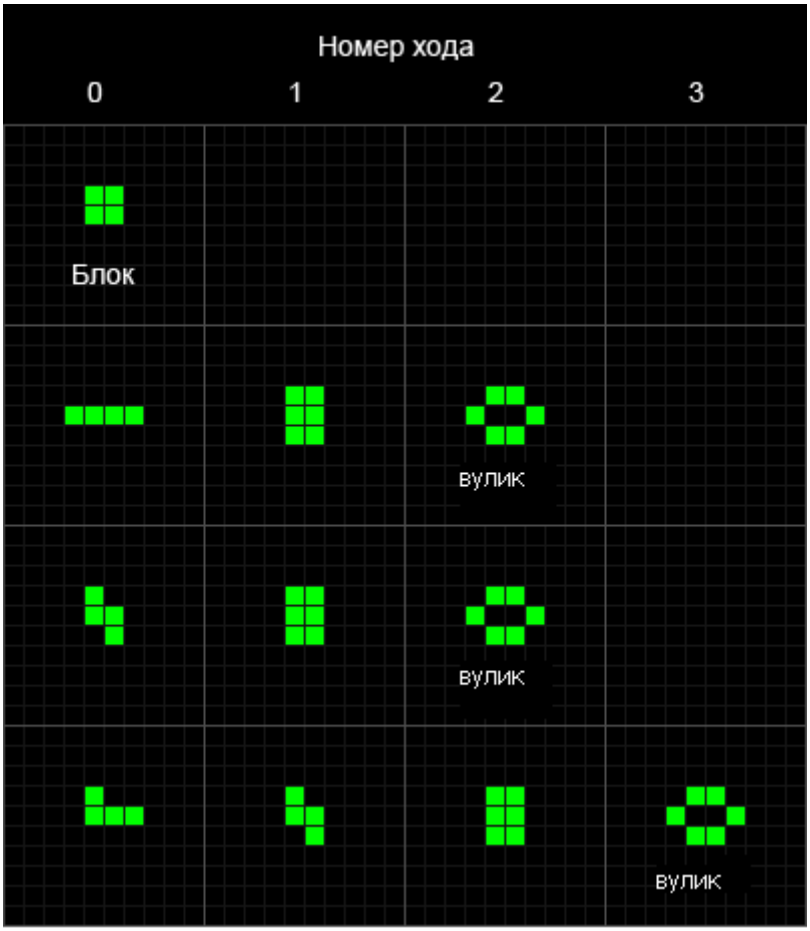


Рисунок А7.6 – Приклади комбінацій з чотирьох фішок

Контрольні питання:

1. Що таке клітинний автомат?
2. Що собою представляє клітинний автомат "Гра життя"? Які його правила?
3. Які типи фігур існують у клітинному автоматі "Гра життя"?
4. Що представляють собою фігури планери у "Грі життя"?
5. Що таке "Едемський сад" у "Грі життя"?
6. Як на основі клітинних автоматів реалізувати багатоагентну інтелектуальну систему?

Список літератури

1. Болотова Л. С. Системы искусственного интеллекта: модели и технологии, основанные на знаниях / Л. С. Болотова – Москва: «Финансы и Статистика», 2012. – 663
2. Терано Т. Прикладные нечеткие системы. : Пер. с яп. / Терано Т., Асаі К., Сугено М., Чернишова Ю. М.: «Мир» - Москва, 1993. 184 с.: іл.
4. Нікольський Ю. В. Дискретна математика: підруч. [для студ. вищ. навч. закл.] / Ю. В. Нікольський, В. В. Пасічник, Ю. М. Щербина – Київ : «Видавнича група BHV», 2007. – 368 с.: іл.
5. Глибовець М.М., Олецький О.В. Штучний інтелект: Підручник для студентів, що навчаються за спец. "Комп’ютерні науки" та "Прикладна математика". – К.: Вид. дім "КМ Академія", 2002. – 366 с.
6. Триумфгородских М.В. Дискретная математика и математическая логика для информатиков, єкономистов и менеджеров. Учебное пособие для ВУЗов. – М.: Диалог-МИФИ, 2011. –180 с.
7. Нікольський Ю.В., Пасічник В.В., Щербина Ю.М. Дискретна математика. Підручник для вищих навчальних закладів. – К.: Видавнича група BHV, 2007. – 368 с.
8. Лямец В.И., Успенко В.И. Основы общей теории систем и системный анализ. Учебное пособие – Харьков: «БУРУН и К», Киев: ООО «КНТ», 2015. – 304 с.