УДК 538.913

Н.В. Юркович, О.В. Герасімов, В.М. Юркович, М.І. Мар’ян
Ужгородський національний університет, 88000, Ужгород, вул. Волошина, 54
e-mail: yurkovich@ukr.net

КОМПОЗИЦІЯ НЕЙРОННИХ МЕРЕЖ З АЛГОРИТМАМИ ХЕББА ТА ПРЯМОГО ПОШИРЕННЯ В СИСТЕМАХ СИМВОЛЬНОГО КОДУВАННЯ

Досліджено нейронні мережі з алгоритмами самоорганізації Хебба і прямого поширення, розглянуто способи їх навчання та можливість використання в системах технічного захисту інформації. Реалізована самоузгоджена модель композиції нейронної мережі з алгоритмом Хебба та прямого поширення, яка здатна розпізнавати зображення символьного кодування даних. Розроблена програма на мові С++ для відповідного розпізнавання коду доступу і створено модуль реалізації алгоритму блокування системи при досягненні попередньо заданого відсотка розпізнавання.

Ключові слова: нейронні мережі, алгоритм Хебба, алгоритм прямого поширення, символьне кодування, електронний цифровий підпис та біометрія, системи захисту інформації, мова програмування С++.

Вступ

Інтелектуальні системи на основі штучних нейронних мереж дозволяють з успіхом вирішувати проблеми розпізнавання образів, виконання прогнозів, оптимізації, асоціативної пам'яті і керування. Традиційні підходи до рішення цих проблем не завжди надають необхідну гнучкість. Багато додатків виграють від використання нейромереж. Штучні нейромережі є електронними моделями нейронної структури мозку, який головним чином навчається з досвіду. Природній аналог доводить, що множина проблем, які поки що не підвладні розв'язуванню наявними комп'ютерами, можуть бути ефективно вирішені блоками нейромереж [1-3].

Тому метою роботи є дослідження нейронних мереж з алгоритмами Хебба і прямого поширення в системах технічного захисту інформації та створення композиції нейронних мереж для відтворення символьного кодування даних на мові програмування С++.

Навчання нейронних мереж за допомогою алгоритму Хебба

Алгоритм навчання Хебба в контексті штучних нейронних мереж можна сформулювати у вигляді таких тверджень [3]:

- якщо два нейрони, пов'язані синаптичним зв'язком, збуджуються синхронно, то відповідна синаптична вага зростає;
- якщо два нейрони по обидві сторони синапсу збуджуються асинхронно, такий синапс слабшає (синаптична вага зменшується).

Можна сформулювати такі властивості синапсу Хебба [3]:

- *Залежність від часу*. Зміна синаптичної ваги залежить від точного часу виникнення передсинаптичного і постсинаптичного сигналів.
- *Локальність*. На зміну синаптичної ваги чинять дію сигнали, що знаходяться в просторово-часовій близькості.
- *Інтерактивність*. Зміна синаптичної ваги визначається сигналами на обидвох його кінцях.
- *Кореляція*. Механізм зміни синаптичної ваги визначається наявністю кореляції між передсинаптичним і постсинаптичним сигналом.

Метод навчання Хебба полягає в зміні ваг за таким правилом [4-9]:

$$W_{ij}(t) = W_{ij}(t-1) + \alpha y_i^{(n-1)} y_i^n \qquad (1)$$

де $y_i^{(n-1)}$ - вихідне значення нейрона і шару $n-1$, y_i^n - вихідне значення нейрона j шару n, $W_{ij}(t)$ і $W_{ij}(t-1)$ - ваговий коефіцієнт синапсу, що з'єднує ці нейрони, на ітераціях n і $n-1$ відповідно; α - коефіцієнт швидкості навчання. Тут і далі під n мається на увазі довільний шар мережі. При навчанні за цим методом посилюються зв'язки між збудженими нейронами.

Існує також і диференціальний метод навчання Хебба [10]:

$$W_{ij}(t) = W_{ij}(t-1) + \alpha\left[y_i^{n-1}(t) - y_i^{(n-1)}(t-1)\right]\left[y_j^{(n)}(t) - y_j^{(n)}(t-1)\right] \qquad (2)$$

Тут $y_i^{(n-1)}$ - вихідне значення нейрона i шару $n-1$ відповідно на ітераціях t і $t-1$, y_i^n - те ж саме для нейрона j шару n. Як видно з формули (2), найефективніше навчаються синапси, що з'єднують ті нейрони, виходи яких найбільш динамічно змінилися в бік збільшення.

Повний алгоритм навчання із застосуванням вищенаведених формул буде виглядати так [6, 9-11]:

1. На стадії ініціалізації всім ваговим коефіцієнтам привласнюються невеликі випадкові значення.

2. На входи мережі подається вхідний образ, і сигнали збудження поширюються по всім шарам згідно принципам класичних прямопоточних (feedforward) мереж, тобто для кожного нейрона розраховується зважена сума його входів, до якої потім застосовується активаційна (передатна) функція нейрона, в результаті чого виходить його вихідне значення y_i^n, $i = 0..M_i - 1$, де M_i - число нейронів у шарі i, $n = 0 \ldots N-1$, а N - число шарів у мережі.

3. На підставі отриманих вихідних значень нейронів за формулою (1) або (2) виробляється зміна вагових коефіцієнтів.

Слід зазначити, що вигляд відгуків на кожен клас вхідних образів не відомий заздалегідь і буде довільним поєднанням станів нейронів вихідного шару, обумовлене випадковим розподілом ваг на стадії ініціалізації. Разом з тим, мережа здатна узагальнювати схожі образи, відносячи їх до одного класу. Тестування навченої мережі дозволяє визначити топологію класів у вихідному шарі. Для приведення відгуків навченої мережі до зручного поданням можна доповнити мережу одним шаром, який, наприклад, за алгоритмом навчання одношарового перцептрона необхідно змусити відображати вихідні реакції мережі в необхідні образи [2].

Символьне кодування з алгоритмами Хебба та прямого поширення. Розробка програми в середовищі C++

На основі співвідношення (1)-(2) була створена програма на мові C++. Її призначенням є розпізнавання образів цифр та латинських букв. Реалізована нейронна мережа в якості передатної функції використовує бінарну порогову функцію (0 і 1).

Інтерфейс програми зображений на рис.1.

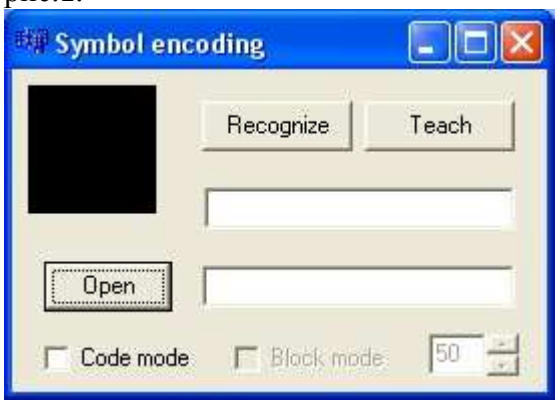

Рис.1. Візуальний інтерфейс програми

Підтримуються зображення формату *.bmp та розміром 64x64 пікселі. OpenPicture відкриває вікно провідника

(рис.2) та дозволяє вибрати зображення з жорсткого диску для розпізнавання.

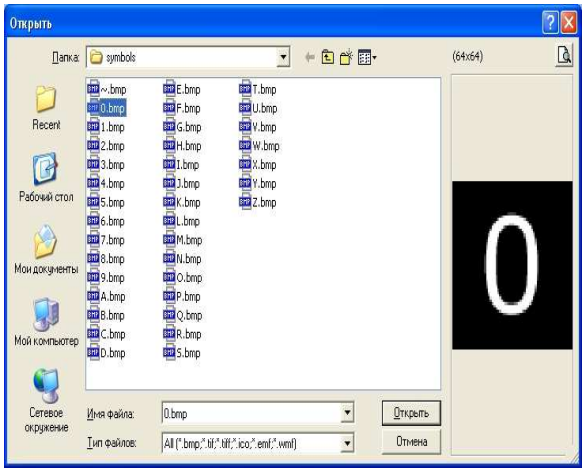

Рис.2. Вікно вибору зображення для розпізнавання

Компонент Recognize викликає алгоритм розпізнавання зображення та вивід результату в поле SymbolEdit. Компонент SymbolEdit використовується для зворотного зв'язку з користувачем. Він виводить результати обчислень (в тому числі і помилки). Також він використовується в процесі навчання як поле вводу символу, що зображений на малюнку. Компонент Teach викликає алгоритм навчання нейронної мережі. Це дозволяє налаштувати мережу таким чином, щоб вона виводила очікувані результати при обчисленнях. Для спрацювання алгоритм потребує зображення символу та ручне введення правильної відповіді в поле SymbolEdit.

Нейронна мережа моделюється двома класами: RecognizeNeuron і RecognizePerceptron. Окрім них є також ще два класи: CodeNeuron і CodePerceptron. Вони є нейронною мережею прямого поширення і складають другу частину програми.

Клас RecognizeNeuron. Цей клас є програмним представленням штучного нейрона. Конструктор цього класу задає початкові вагові коефіцієнти і крок їх зміни при навчанні. Ваги синапсів нейрона зберігаються в його атрибуті weights[]. Вхідні дані є масивом пікселів, перемножаються на відповідний кожному пікселю ваговий коефіцієнт, після чого отримані добутки сумуються. Це реалізовується методом activate(int[]). В результаті роботи нейрона на виході отримується певна сума синаптичних ваг, яка характеризує збудження нейрона на зображення. Нейрон з найбільшим збудженням визначає результат розпізнавання.

Навчання нейрона реалізується методом changeWeights(int[]). На вхід подається масив пікселів зображення, якому буде відповідати даний нейрон. Він порівнюється із масивом вагових коефіцієнтів нейрона. Якщо піксель зафарбований, коефіцієнт збільшується на значення step. Атрибут totalWeights зберігає фіксовану суму всіх ваг. Він використовується для того, щоб вирахувати, на яке значення мають зменшитись інші вагові коефіцієнти. Такі маніпуляції з коефіцієнтами необхідні для правильного розпізнавання спотворених зображень.

Клас RecognizePerceptron. Реалізовує нейрону мережу. Складається з масиву нейронів neurons[]. Метод recognize(int[]) подає вхідний масив пікселів кожному нейрону і порівнює їх рівень збудження. Індекс нейрона з найбільшим збудженням і є результатом розпізнавання (мережа працює не з самим символом, а з його кодом; індекс кожного нейрона відповідає певному символу).

Навчання мережі реалізовано методом teach(int[],int). Він отримує на вхід масив пікселів зображення та відповідний йому символ, вказаний користувачем. Цей масив передається в метод changeWeights(int[]) нейрона з порядковим номером, що відповідає коду вказаного символу.

Тестування та аналіз отриманих результатів

На момент запуску програми нейронна мережа ще не вміє розпізнавати образи символів. Для того, щоб навчити нейронну мережу розпізнавати, необхідно в поле SymbolEdit ввести символ, що зображений на малюнку, і натиснути кнопку Teach. В результаті нейронна мережа запам'ятає символ (створить внутрішню асоціацію між розташуванням

пікселів на зображенні та введеним символом) (рис.3).

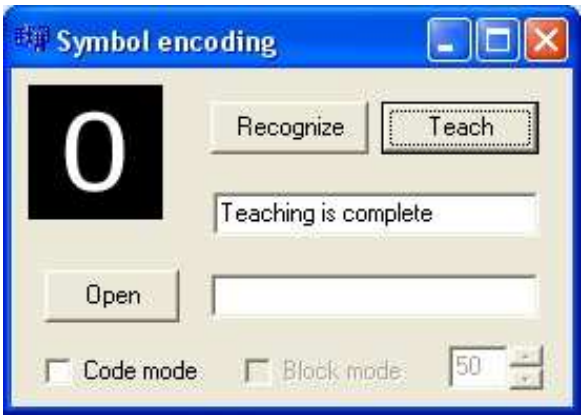

Рис.3. Процес розпізнавання символів нейронною мережею

Символ, введений користувачем, перетворюється у свій чисельний еквівалент у відповідності зі стандартом ANSI. Потім отримане число проходить певні перетворення і приймає значення від 0 до 35 (10 цифр і 26 літер). Ці перетворення потрібні через те, що в інтервалі до літери Z окрім цифр та літер присутні ще й символи керування та розділові символи. Нейронна мережа складається із 36 нейронів, кожен із яких відповідає за свій символ. Під час навчання навчається лише нейрон із відповідним індексом. Якщо для навчання мережі ввести в поле SymbolEdit декілька символів або символ, що не підтримується програмою (не цифру і не латинську букву), нейронна мережа повідомить про некоректність даних для навчання (рис.4).

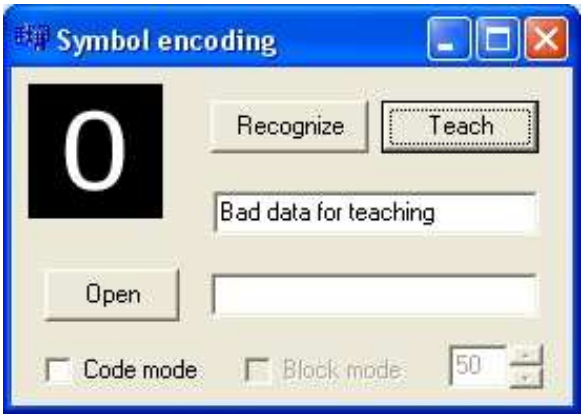

Рис.4. Результат виклику алгоритму навчання внаслідок некоректного введення вхідних даних

Під час навчання зображення розбивається на масив пікселів. Кожен піксель містить інформацію про свій колір (шестизначне шістнадцяткове число). Цей масив переписується таким чином, щоб всі кольори, відмінні від 0 (код чорного кольору за замовчуванням), розглядались як 1. Таким чином фоном є чорний колір, а символ може бути зображений будь-яким іншим кольором.

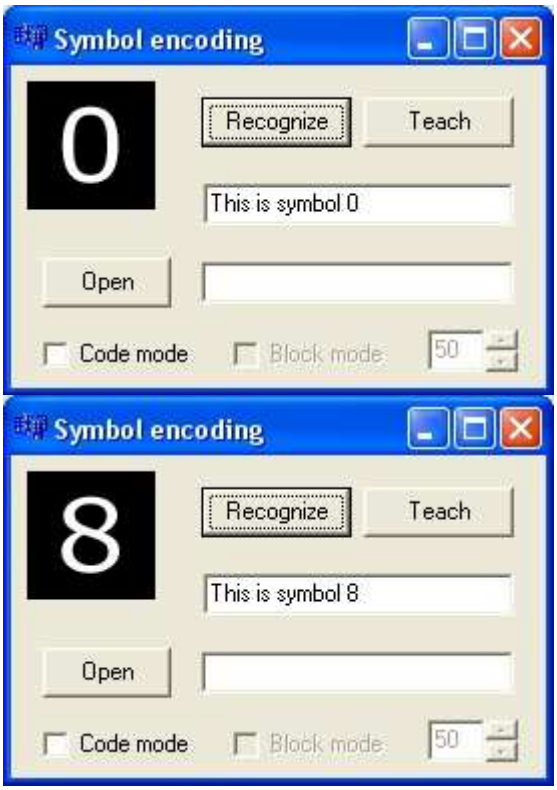

Рис.5. Результат роботи нейронної мережі після навчання

Після проходження навчання для кожного символу, нейронна мережа стає здатною розпізнавати зображення (рис.5).

Блокування символьного кодування та декодування даних з використанням композиції нейронних мереж

Окремим модулем реалізовано алгоритм введення коду доступу та блокування роботи програми при певних попередньо заданих умовах. Він підключається перемикачем CodeCheckBox. В результаті принцип роботи програми змінюється. Стають доступні компоненти CodeEdit і BlockCheckBox. Кнопка Teach блокується,

так як в цьому режимі вона непотрібна. Поле SymbolEdit стає недоступним для редагування – тут виводиться процес вводу коду користувачем.

Ідея модуля полягає в тому, що користувач намагається ввести код доступу (звичайний пароль, відбиток пальця, електронна картка тощо [10,12]). Програма порівнює введені дані із еталоном. Якщо програма працює у режимі блокування (вмикається перемикачем BlockCheckBox), то при досягненні певного відсотка розпізнавання вхідних даних (тобто співпадіння поданих через зображення символів із еталоном) програма блокується. Таким чином, відбувається захист від несанкціонованого доступу зловмисником.

Реалізована програма є одним із варіантів вищенаведених випадків. У поле CodeEdit вводиться код, що складається з цифр та латинських літер. Це є еталон, з яким будуть порівнюватись дані, введені користувачем через кнопку OpenPicture (тобто вводиться зображення певного символу). Поле SymbolEdit автоматично заповнюється символами «*». Їх кількість рівна кількості символів у еталоні. При натисканні кнопки Recognize зображення розпізнається у відповідності із навчанням, пройденим в першому режимі роботи програми (перемикач CodeCheckBox вимкнений). Якщо подане на вхід зображення містить символ, вказаний у еталоні, то поле SymbolEdit поновлюється і у відповідне місце записується введений символ (рис.6).

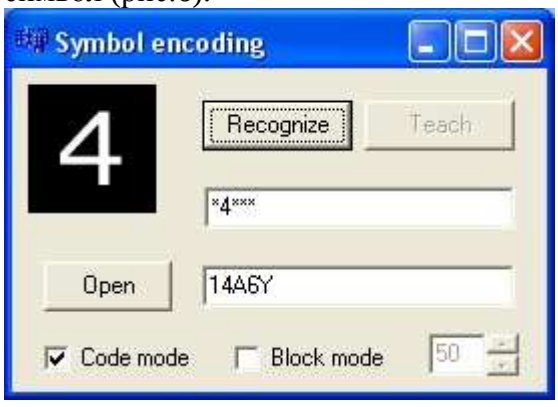

Рис.6. Введення коду доступу користувачем

Таким чином імітується процес ідентифікації користувача.

Робота програми може бути переведена в режим блокування доступу при досягненні певного відсотка розпізнавання. Це досягається вмиканням перемикача BlockCheckBox. Завдяки цьому стають доступними поле BlockEdit і стрілки BlockUpDown. Вони використовуються для задання граничного значення розпізнавання у відсотках. Тепер при досягненні вказаного відсотка розпізнавання, програма заблокується (рис.7).

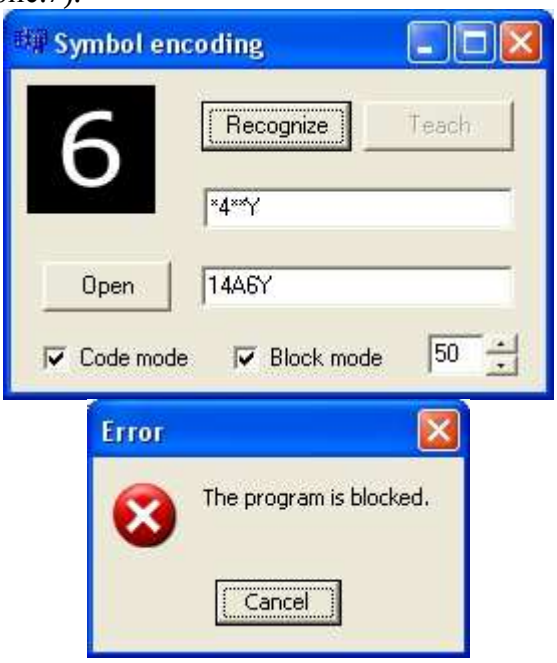

Рис.7. Блокування роботи програми при досягненні вказаного відсотка розпізнавання

Програмно режим коду та блокування реалізований класами CodeNeuron і CodePerceptron.

Клас CodeNeuron. Цей нейрон зберігає певне еталонне значення в атрибуті value. При виклику функції activate(int) нейрон порівнює вхідне значення із еталонним. Метод getWeights() повертає значення вагового коефіцієнта.

Навчання нейрона реалізується методами setValue(int) і changeWeights(int). Перший задає еталонне значення для нейрона, а другий змінює ваговий коефіцієнт на 1 або -1 (кожен нейрон відповідає за окремий символ коду, значення -1 встановлюється коли нейрон вже спрацював, тобто він вже не приймає участі в подальшій роботі так як відповідний йому символ вже введений користувачем).

Клас CodePerceptron. Містить масив нейронів neurons[]. Метод compare(int) приймає на вхід символ, введений користувачем як частина коду. Далі викликаються методи activate(int) кожного нейрона. Якщо якийсь з них підтвердив співпадіння, то мережа повертає індекс цього нейрона (тобто порядковий номер символу в коді, з яким співпав введений користувачем символ) і змінює його ваговий коефіцієнт методом changeWeights(int,int) на -1. Метод teach(int,int) задає еталон окремого нейрона.

Алгоритм блокування реалізується методом blocked(int). На вхід він приймає граничне значення відсотка розпізнавання, яке задається користувачем. Цей метод дістає значення вагових коефіцієнтів всіх нейронів і вираховує, скільки з них мають коефіцієнт рівний 1. Визначає їх відсоток і порівнює із заданим граничним значенням.

Висновки

На основі аналізу літературних даних теорії нейронних мереж та прикладів їх програмних реалізацій показана можливість їх застосування в системах технічного захисту інформації. Розроблена програма для символьного розпізнавання з використанням мережі Хебба та прямого поширення на мові С++. Проведено тестування та діагностика моделі, а також розроблено інтерфейс. Реалізована модель нейронної мережі здатна розпізнавати символи на зображенні – цифри та латинські букви. Створено модуль реалізації алгоритму розпізнавання послідовності символів і їх співставлення з конкретним паролем, імітація отримання доступу, та реалізована можливість блокування роботи програми при досягненні заданого відсотка розпізнавання.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Саймон Хайкин. Нейронные сети. - М.: ООО «И.Д. Вильямс», 2006. – 1104с.
2. Мар’ян М.І., Юркович Н.В. Дисипативні структури та процеси самоорганізації в некристалічних матеріалах. Науковий вісник Ужгородського університету. Серія Фізика. 2011.-№29.-с.79-86.
3. Каширина И.Л. Искусственные нейронные сети: Учебное пособие. – Воронеж: Изд-во ВГУ, 2005. – 51с.
4. Круглов В.В., Борисов В.В. Искусственные нейронные сети. Теория и практика. – – М.: Горячая линия – Телеком, 2002. – 382с.
5. Ясницкий Л.Н. Введение в искусственный интеллект.– М.: Издательский центр «Академия», 2008. – 176с.
6. Минский М., Пейперт С. Персептроны. – М.: Мир, 1971. – 261с.
7. Уоссермен Ф. Нейрокомпьютерная техника. – М.: Мир, 1992. – 240с.
8. Роберт Каллан. Основные концепции нейронных сетей: пер. с англ. – М.: ООО «И.Д. Вильямс», 2001. – 287с.
9. Аксенов С.В., Новосельцев В.Б. Организация и использование нейронных сетей (методы и технологии) / Под общ. ред. В.Б. Новосельцева. – Томск: Изд-во НТЛ, 2006. – 128с.
10. Пашко П.В. Митні інформаційні технології. – К.: Знання, 2011. – 391с.
11. Кудрявцева С.П., Колос В.В. Міжнародна інформація.– К.: Видавничий Дім «Слово», 2005. – 400с.
12. Горбань А.Н. Обучение нейронных сетей. – М.: ПараГраф, 1990. – 159с.

Стаття надійшла до редакції 16.10.2014

Yurkovych N.V., Herasimov O.V., Yurkovych V.M., Mar'yan M.I.
Uzhhorod National University, 88000, Uzhhorod, Voloshin Str., 54
e-mail: yurkovich@ukr.net

COMPOSITION OF NEURAL NETWORKS BY HEBB ALGORITHM AND DIRECT SPREADING IN CHARACTERS ENCODING SYSTEMS

Investigated neural network by Hebb self-organizated algorithms and direct spreading, discussed ways of learning it and ability to use it at systems of technical information protection. Implemented self-consistent model of composition of the neural network by Hebb algorithm and direct spreading, which is able to detect image data character encoding. The program developed in C ++ for appropriate recognition code access, also created the module of the algorithm realisation blocking system upon reaching a preset percentage of recognition.

Keywords: neural networks, by Hebb algorithm, algorithm direct spreading, character encoding, digital signature and biometrics, information security systems, the programming language C ++.

Юркович Н.В., Герасимов О.В., Юркович В.М., Марьян М.И.
Ужгородский национальный университет, 88000, Ужгород, ул. Волошина, 54
e-mail: yurkovich@ukr.net

КОМПОЗИЦИЯ НЕЙРОННЫХ СЕТЕЙ С АЛГОРИТМАМИ ХЕББА И ПРЯМОГО РАСПРОСТРАНЕНИЯ В СИСТЕМАХ СИМВОЛЬНОГО КОДИРОВАНИЯ

Исследованы нейронные сети с алгоритмами самоорганизации Хебба и прямого распространения, рассмотрены способы их обучения и возможность использования в системах технической защиты информации. Реализована самосогласованная модель нейронной сети с алгоритмами Хебба и прямого распространения, которая способна распознавать изображения символьного кодирования данных. Разработана программа на языке С++ для соответствующего распознавания кода доступа и создано модуль реализации алгоритму блокирования системы при достижении предварительно заданного процента распознавания.

Ключевые слова: нейронные сети, алгоритм Хебба, алгоритм прямого распространения, символьное кодирование, электронная цифровая подпись и биометрия, системы защиты информации, язык программирования С++.