УДК 004.032.2

Я. С. Парамуд, В. І. Яркун
Національний університет “Львівська політехніка”,
кафедра електронних обчислювальних машин

АЛГОРИТМІЧНО-ПРОГРАМНІ ЗАСОБИ РОЗПІЗНАВАННЯ РУКОПИСНИХ СИМВОЛІВ НА ЗОБРАЖЕННІ

Розглянуто алгоритмічно-програмні засоби розпізнавання рукописних символів на зображенні за алгоритмом логістичної регресії та побудови штучної нейронної мережі (ШНМ). Здійснено порівняльний аналіз цих двох підходів. Виконано тестування рукописних цифр. Встановлено, що краща якість розпізнавання досягається у разі використання штучної нейронної мережі.

Ключові слова: логістична регресія, штучна нейронна мережа, розпізнавання символів, машинне навчання, функція вартості, градієнт пониження.

Y. Paramud, V. Yarkun
Lviv Polytechnic National University,
Computer Engineering Department

ALGORITHMIC AND SOFTWARE MEANS OF HANDWRITTEN SYMBOLS RECOGNITION

In this article is considered the algorithm of logistic regression and construction of the neural network for the recognition of handwritten symbols in the image. Examples of implementation of two approaches for solving the problem of numerical recognition are given. The efficiency of using a neural network, as the provision of the most reliable recognition results, is explored.

Key words: logistic regression, neural network, symbols recognition, machine learning, cost function, gradient descent.

Вступ

Засоби автоматизованого уведення текстової інформації є одними із найпоширеніших периферійних пристроїв комп’ютерних систем. Основним призначенням таких пристроїв є читання текстових документів, розпізнавання отриманої інформації, перетворення прочитаних кодів на коди символів за прийнятим способом кодування та передавання отриманих кодів до ядра комп’ютера. До пристроїв автоматизованого читання текстових документів можна зарахувати сканери та відеокамери. Вони переважно перетворюють текстову інформацію на матрицю кодів певної розрядності для визначеної кількості рядків, стовпців. Ця задача розв’язується достатньо якісно. Складнішою є задача розпізнавання прочитаних символів. Задача істотно ускладнюється для рукописних текстів. Відповідно актуальними є дослідження в царині алгоритмічно-програмних засобів розпізнавання символів у разі автоматизованого уведення у комп’ютер рукописних текстових документів.

Окреслення проблеми

Розпізнавання символів у разі автоматизованого уведення у комп'ютер текстових документів вважають одним із різновидів розпізнавання образів, що зараховують до фундаментальних проблем систем штучного інтелекту [1]. Задача розпізнавання образів має велике практичне значення. Замість терміна "розпізнавання" часто використовується інший термін – "класифікація". Ці два терміни у багатьох випадках розглядаються як синоніми, але не є повністю взаємозамінними. Кожний з цих термінів має свої сфери застосування й інтерпретація обох термінів часто залежить від специфіки конкретної задачі.

Задачі розпізнавання в багатьох випадках розв'язують люди, причому роблять це, як правило, підсвідомо. Однак спроби побудувати штучні системи розпізнавання не настільки переконливі. Основна проблема полягає у тому, що здебільшого неможливо адекватно визначити ознаки, на основі яких слід здійснювати розпізнавання. Для задач, для яких такі ознаки вдається виділити, штучні системи розпізнавання набули значного поширення і широко використовуються [1].

Аналіз останніх досліджень та публікацій

Алгоритм логістичної регресії є одним із базових алгоритмів для задач розпізнавання. Він широко застосовується, коли постає задача класифікації. Цей алгоритм дає змогу розподілити множину об'єктів за певними їхніми ознаками та відповідно їх класифікувати. Математична модель нейронної мережі, а також її програмна та апаратна реалізація застосовуються у задачах прогнозування, для розпізнавання образів, у задачах керування. Традиційні програми не забезпечують необхідною гнучкістю і багато прикладних задач ефективно розв'язують, використовуючи штучні нейронні мережі (ШНМ). Такі мережі є спрощеними електронними моделями нейронної структури мозку, який, переважно, розв'язує прикладні задачі за отриманим досвідом. Практичний досвід доводить, що багато задач, які складно розв'язати традиційними комп'ютерами, можна ефективно розв'язати за допомогою нейромереж [2].

Сьогодні є доволі багато підходів для розв'язання задачі розпізнавання символів на зображенні, проте велика частина з них надає результати недостатньої вірогідності з великими відсотками помилок розпізнавання, що потребує додаткових досліджень та вдосконалень алгоритмів.

ШНМ являють собою систему з'єднаних між собою простих обробників (штучних нейронів), які взаємодіють. Такі обробники зазвичай доволі прості (особливо порівняно з процесорами, що застосовують у персональних комп'ютерах). Кожен обробник такої мережі має справу лише з сигналами, які він періодично отримує, і сигналами, які він періодично надсилає іншим обробникам. І тим не менш, з'єднані в достатньо велику мережу з керованою взаємодією, такі локально прості обробники разом здатні виконувати доволі складні завдання [3]. Відповідно перед виконанням складного завдання кожний обробник має бути налаштованим (навченим) на виконання конкретного простого завдання.

ШНМ можна характеризувати певними загальними особливостями [4, 5]. У царині обчислювальної техніки та програмування нейронну мережу можна розглядати як засоби розв'язання задачі ефективного паралелізму. В царині практичного застосування нейронна мережа найчастіше використовується в задачах адаптивного керування та в робототехніці. Процес навчання нейронної мережі розглядається як багатопараметрична задача нелінійної оптимізації, є окремим випадком методів розпізнавання образів, дискримінантного аналізу, методів кластерування.

У літературних джерелах запропоновано два визначення машинного навчання [4, 5]. За першим це "Область навчання, яка дає комп'ютерам можливість навчитися без явного програмування". Таке визначення вважається дещо застарілим. Сучасніше визначення: "Комп'ютерна програма навчається з досвіду E щодо певного класу завдань T та показника продуктивності P, якщо його ефективність при завданнях T, виміряна за допомогою P, покращується з досвідом E". Тобто машинне навчання досліджує вивчення та побудову алгоритмів, які можуть навчатися з даних та виконувати передбачуваний аналіз на них.

Аналіз останніх досліджень та публікацій вказує: все ще недостатньо уваги приділено застосуванню ШНМ у задачах розпізнавання рукописних символів у разі автоматизованого уведення у комп'ютер текстової інформації.

Мета статті

Мета цієї роботи – розробити і дослідити алгоритмічно-програмні засоби для розпізнавання рукописних символів на зображенні, продемонструвати приклади реалізації вищезгаданих алгоритмів та надати результати, які показують якість розпізнавання.

Основний матеріал дослідження

За результатами аналізу останніх досліджень та публікацій можна зробити висновок, що до найперспективніших підходів щодо розв'язання задач розпізнавання рукописних символів у разі автоматизованого уведення у комп'ютер текстової інформації належить використання алгоритму логістичної регресії та ШНМ. В обох підходах важливим є машинне навчання.

Задача машинного навчання, а саме навчання з учителем (supervised learning), коли система навчається за допомогою наявної множини прикладів, описується так: метою є, з урахуванням навчального набору, навчити функцію h : X → Y так, що h (x) є "добрим" провісником для відповідних значень Y. З історичних причин ця функція h називається гіпотезою. Відображення цього процесу подано на рис. 1.

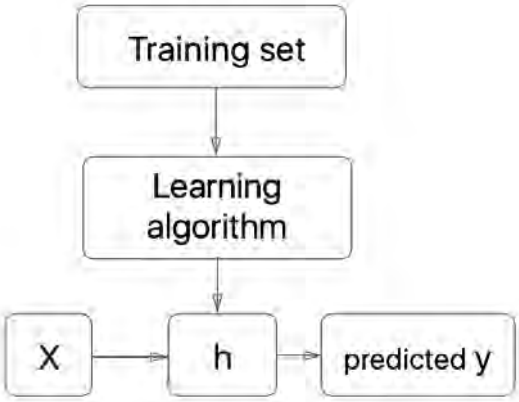

Рис. 1. Узагальнена схема використання гіпотези

Можна виміряти точність функції гіпотези (h), використовуючи функцію вартості (cost function), яка позначається J (θ), де θ являє собою особливості (features), які беруть до уваги під час обчислення функції (h). Аналітичні особливості машинного навчання за алгоритмом логістичної регресії такі [5].

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\text{Cost}(h_\theta(x), y) = -\log(h_\theta(x)) \qquad \text{if y} = 1$$
$$\text{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \qquad \text{if y} = 0$$

$$\text{Cost}(h_\theta(x), y) = 0 \text{ if } h_\theta(x) = y$$
$$\text{Cost}(h_\theta(x), y) \to \infty \text{ if } y = 0 \text{ and } h_\theta(x) \to 1$$
$$\text{Cost}(h_\theta(x), y) \to \infty \text{ if } y = 1 \text{ and } h_\theta(x) \to 0$$

Функцію Cost спрощено до такого вигляду:

$$\text{Cost}(h_\theta(x), y) = -y \log(h_\theta(x)) - (1 - y)\log(1 - h_\theta(x))$$

Отже, тепер можна повністю записати функцію J:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$$

Векторизована форма матиме такий вигляд:

$$h = g(X\theta)$$

$$J(\theta) = \frac{1}{m}\cdot\left(-y^T\log(h) - (1-y)^T\log(1-h)\right)$$

Де відповідно функція h дорівнює:

$$h_\theta(x) = g(\theta^T x)$$

$$z = \theta^T x$$

$$g(z) = \frac{1}{1+e^{-z}}$$

Далі необхідна мінімізація функції J, це можна зробити за допомогою алгоритму градієнта пониження (gradient descent), далі використовуватиметься позначення GD.

Загальне подання алгоритму GD таке:

$$Repeat\ \{$$

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta)$$

$$\}$$

У разі обчислення похідної GD буде таким:

$$Repeat\ \{$$

$$\theta_j := \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

$$\}$$

Векторизована форма алгоритму GD:

$$\theta := \theta - \frac{\alpha}{m}X^T(g(X\theta) - \vec{y})$$

Програмна реалізація алгоритму

Програмну реалізацію продемонстрували, використовуючи середовище Matlab. Для тренування необхідно взяти **m** прикладів. Дослідження виконано для **m** = 5000. Така кількість прикладів рукописних цифр, згідно з аналітичними висновками, буде достатньою для отримання достовірних результатів. Ці приклади зберігаються у файлі dataset.mat (формат .mat означає, що дані були збережені у матричному вигляді середовищем Matlab). Кожен приклад являє собою чорно-біле зображення у вигляді матриці **a x b** пікселів (розмір може бути довільним), у цьому випадку взято 20 x 20 пікселів – для цього дослідження вибрано саме такий розмір зображення для розпізнавання. Для відповідних задач та вимог може бути вибраний довільний розмір зображення. Кількість прикладів має бути достатньо великою, щоб гарантувати високу достовірність отриманих результатів. Для кольорових зображень необхідно враховувати кількість розрядів, якими кодується колір у кожному пікселі. Усі ці фактори потрібно враховувати під час попереднього налаштування та розпізнавання зображення. Кожен піксель являє собою число, яке вказує на інтенсивність сірого кольору (вибрано зображення чорно-білого кольору) в цій позиції. **a x b** матрицю можна "розгорнути" в **a * b**-вимірний вектор і звідси отримати матрицю прикладів **X** розмірністю **m x (a*b).** У цьому дослідженні матриця 20 x 20 пікселів "розгорнута" в 400-вимірний вектор. Отже,

отримуємо матрицю прикладів **X** розмірністю 5000 x 400, де кожен рядок являє собою зображення рукописної цифри:

$$X = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(m)})^T — \end{bmatrix}$$

Інша частина тренувальних прикладів містить вектор **y**, розмірністю **m x q**, де **q** містить розмір, необхідний для надання правильної відповіді під час тренування, у цьому прикладі **y** є 5000 x 1 (цифра 1 означає, що правильна відповідь для певного прикладу із 5000 тільки одна – і це цифра від 0 до 9), який відповідає рукописним цифрам у матриці **X**. 100 прикладів зображень з рукописними цифрами відображено на рис. 2.

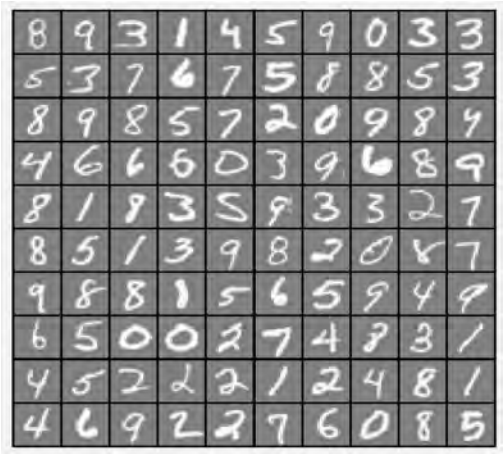

Рис. 2. Приклади зображень

Функція вартості для логістичної регресії реалізована в програмному модулі 1.

```
function [J, grad] = lrCostFunction(theta, X, y, lambda)

m = length(y); % number of training examples

J = 0;
grad = zeros(size(theta));

tempTheta = theta(2:end)
regParam = lambda * sum(tempTheta .^2)/(2 * m);

h = sigmoid(theta' * X');

J = (-y' * log(h)' - (1 - y)' * log(1 - h)')/m + regParam;

g = sigmoid(X * theta);
temp = theta;
temp(1) = 0;
grad = (X' * (g - y))/m + lambda * temp/m;

grad = grad(:);

end
```

Програмний модуль 1 обчислює функцію вартості та GD для алгоритму логістичної регресії, використовуючи згадані вище формули.

Далі необхідно реалізувати функцію oneVsAll, яка тренує кожен класифікатор для кожного класу цифр, яких всього 10. Ця функція реалізована в поданому нижче програмному модулі 2.

```
function [all_theta] = oneVsAll(X, y, num_labels, lambda)

m = size(X, 1);
n = size(X, 2);

all_theta = zeros(num_labels, n + 1);

X = [ones(m, 1) X];

initial_theta = zeros(n + 1, 1);
options = optimset('GradObj', 'on', 'MaxIter', 50);

temp = zeros(num_labels, n + 1);

for i = 1 : num_labels
    temp(i, :) = (fmincg(@(t)(lrCostFunction(t, X, (y == i), lambda)),
initial_theta, options))'
end

all_theta = temp;

end
```

Функція, яка буде займатись передбаченнями, наведена в програмному модулі 3.

```
function p = predictOneVsAll(all_theta, X)

m = size(X, 1);
num_labels = size(all_theta, 1);

p = zeros(size(X, 1), 1);

% Add ones to the X data matrix
X = [ones(m, 1) X];

 [pval, predictions] = max(sigmoid(X * all_theta'), [], 2)

% Convert to ascii and return the result

character = num2str(predictions)
p = double(character)

end
```

Точність розпізнавання, якщо використовують програмний модуль 3, становить 95.08 % – це середня обчислена точність розпізнавання зображень, які подаються у функцію predictOneVsAll.

Наступний підхід розв’язання задачі розпізнавання – використовуючи штучну нейронну мережу. Застосовуючи цей підхід, необхідно мати натреновані ваги для відповідного шару. Для тренування ваг можна використати алгоритм “Gradient descent”(GD), описаний вище. Його слід застосувати після виконання алгоритму “Backpropagation”, який мінімізує вибрану “Cost function” для відповідної задачі. Загальну схему підходу з ШНМ зображено на рис. 3.

Функції “Cost” та “Gradient descent” продемонстровано у реалізації підпрограми 1 та формулами, наведеними вище. Алгоритм “Feedforward propagation” реалізований у підпрограмі 4. Вектори ваг можна визначити за допомогою алгоритму “Backpropagation”, описаного нижче.

“Cost function” нейромережі матиме такий вигляд [6]:

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[y_k^{(i)}\log((h_\Theta(x^{(i)}))_k) + (1-y_k^{(i)})\log(1-(h_\Theta(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{j,i}^{(l)})^2,$$

де K – кількість вихідних вузлів, S_l – кількість вузлів у шарі l, не рахуючи “bias unit”; L – загальна кількість шарів у мережі, λ – параметр регуляризації, визначає, наскільки сильно на розрахунок “Cost function” впливають Θ параметри.

ШНМ міститиме три шари: вхідний, один внутрішній, вихідний, як зображено на рис. 4.

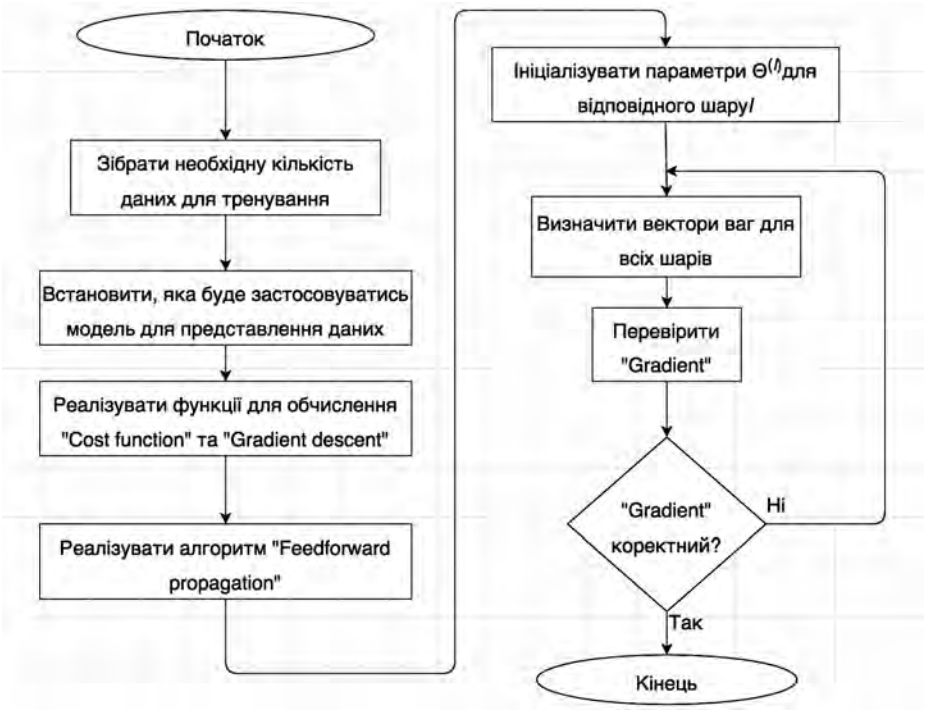

Рис. 3. Загальна схема алгоритму тренування ШНМ

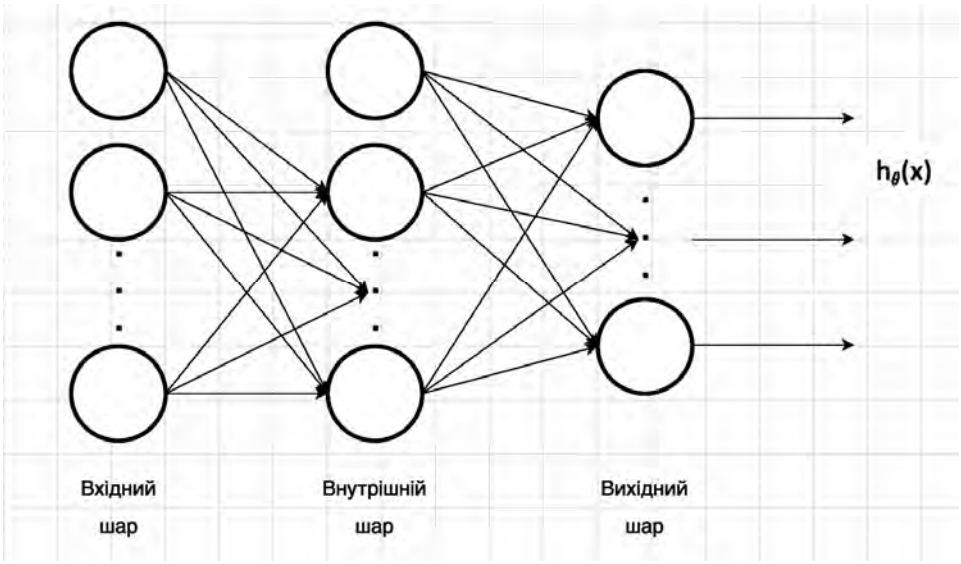

Рис. 4. Структура шарів ШНМ

На рис. 4 зображена структура шарів нейронної мережі, яка є задовільною для заданої задачі. Верхній вузол, у внутрішньому шарі, називається “bias unit”, він не набуває ніяких параметрів і призначений для зручності обислень. Його значення переважно завжди ініціалізовують 1.

Алгоритм “Backpropagation”

Набір тренувань $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$
$\triangle_{ij}^{(l)} = 0$
For $i = 1$ to m
$a^{(1)} = x^{(i)}$

Виконати “Forward propagation”, щоб обчислити $a^{(l)}$ для $l = 2, 3, \ldots, L$

Використовуючи *“y (i)”*, обчислити $\delta^{(L)} = a^{(L)} - y^{(i)}$, де δ – помилка в шарі L

Обчислити $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)}$

$\triangle_{ij}^{(l)} := \triangle_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$, j – лічильник вузла у відповідному шарі

$D_{ij}^{(l)} := \frac{1}{m} \triangle_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)}$ if $j \neq 0 \frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$
$D_{ij}^{(l)} := \frac{1}{m} \triangle_{ij}^{(l)}$ if $j = 0$

Використовуючи алгоритм “Backpropagation”, програма проходитиме від вихідного шару до вхідного та обчислюватиме похідні ваг. У результаті виконання алгоритму “Backpropagation” буде отримано вектор похідних до ваг відповідного шару. Після цього можна застосувати алгоритм GD, безпосередньо для їх розрахунку. Вказані алгоритми реалізовано в поданому нижче програмному модулі 4.

```
function p = predict(Theta1, Theta2, X)
%   p = predict(Theta1, Theta2, X) outputs the predicted label of X given the
%   trained weights of a neural network (Theta1, Theta2)

m = size(X, 1);
num_labels = size(Theta2, 1);

p = zeros(size(X, 1), 1);

X = [ones(m, 1) X];


a1 = sigmoid(X * Theta1');
a1 = [ones(m, 1) a1];

a2 = sigmoid(a1 * Theta2')

[pval, predictions] = max(a2, [], 2)

% Convert to ascii and return the result

character = num2str(predictions)
p = double(character)

end
```

Програмний модуль 4 використовує алгоритм “feedforward propagation” для повернення передбачень з функції predict. Якщо використовувати нейромережу, точність передбачення становить 97.52 %. Точність обчислено із використанням програмного модуля 5.

```
pred = predict(Theta1, Theta2, X);

fprintf('Точність: %f\n', mean(double(pred == y)) * 100);
```

Використовуючи наведені вище програмні модулі, можна розпізнавати “нарізані” символи розміром a x b пікселів, у цьому випадку 20x20 пікселів, з довільного зображення. Ці програмні модулі можна модифікувати для розпізнавання довільних рукописних символів довільного алфавіту.

Висновки

Продемонстровано використання алгоритмічно-програмних засобів для розпізнавання рукописних символів на зображенні. Наведено приклади використання алгоритмів: логістичної регресії та нейронної мережі. Встановлено, що використання ШНМ є ефективнішим (за точністю розпізнавання) для задачі розпізнавання символів. Наведені програмні модулі є універсальними для інтегрування у програмне забезпечення для розпізнавання довільних рукописних символів довільного алфавіту. Запропоновані алгоритмічно-програмні засоби можуть застосовуватися для автоматизованого розпізнавання рукописних символів поштових кодів, розпізнавання сум, написаних на банківських чеках, та в інших прикладних задачах.

1. Expression of images recognition [Electronic resource] / wiki. – Access mode: https://uk.wikipedia.org/wiki/Requirements recognition. 2. Lukin V. E. Analysis of the use of technology of artificial neural networks as a new approach to signal processing / V. Lukin // Telecommunication and information technologies. – 2014 – P. 81–82. 3. Artificial_neuronal_network [Electronic resource] / wiki. – Access mode: https://uk.wikipedia.org/wiki/New_neuronal_network. 4. What is machine learning [Electronic resource] / Coursera. – Access mode: https://www.coursera.org/learn/machine-learning/supplement/aAgxl/what-is-machine-learning 5. Machine learning [Electronic resource] / Coursera. – Access mode: https://www.coursera.org/learn/machine-learning/home/week/3. 6. Backpropagation algorithm [Electronic resource] / Coursera. – Access mode: https://www.coursera.org/learn/machine-learning/supplement/pjdBA/backpropagation-algorithm. 7. How the backpropagation algorithm works [Electronic resource] / Neuralnetworksanddeeplearning. – Access mode: http://neuralnetworksanddeeplearning.com/chap2.html.