УДК 004.02:519.6
DOI https://doi.org/10.26661/2786-6254-2023-2-07

ОПЕРАТОРНИЙ ГЕНЕТИЧНИЙ АЛГОРИТМ І НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ

Олійник Л. О.
кандидат фізико-математичних наук,
доктор філософії, доцент, директор
ВСП «Технологічний фаховий коледж
Дніпровського державного технічного університету»
просп. Конституції, 2А, Кам'янське, Дніпропетровська область, Україна
orcid.org/0000-0002-4392-0048
l.olejnik57@gmail.com

Ключові слова: *інволютивний оператор, стохастичний оператор, кросовер, мутація, фітнес-функція, n-вимірний гіперкуб, бінарний код, код Грея.*

Робота присвячена дослідженню ефективності застосування операторного генетичного алгоритму до навчання нейронних мереж.

Як відомо генетичні алгоритми, різні їхні інтерпретації, є досить ефективним інструментом пошуку розв'язків задач оптимізації. Дія генетичного алгоритму базується на випадковому процесі формування популяції можливих розв'язків, серед яких відбирають найкращі в певному сенсі.

У статті представлено розроблену автором ***операторну модель*** генетичного алгоритму (далі – ***операторний генетичний алгоритм***), яка застосовується до навчання нейронної мережі. Операторний генетичний алгоритм базується на застосуванні інволютивних операторів, що діють у декартовому добутку двох екземплярів n-вимірного евклідового простору і здійснюють операції кросоверу та мутації. Крім того головним інструментом формування популяцій векторів-хромосом є стохастичні оператори, що діють у тому ж декартовому добутку. Результат дії цих операторів можна інтерпретувати як узагальнені кросовер і мутація. Дана операторна модель дозволяє на кожному ітераційному кроці алгоритму формувати в області пошуку популяції векторів-хромосом невеликі за потужністю. Операторна модель генетичного алгоритму передбачає двійкове кодування інволютивних операторів, які перетворюють вершини гіперкубу, що являється областю пошуку, але точки області пошуку представляються у десятковому форматі.

В роботі досліджується ефективність операторного генетичного алгоритму для навчання відомої нейронної мережі, що реалізує булеву функцію XOR. Сенс розгляду цього прикладу полягає у тому, що він є частинним випадком задачі класифікації точок одиничного гіперкуба довільної вимірності. В результаті застосування операторного генетичного алгоритму до навчання нейронної мережі XOR отримано множину операторів, які ефективно (з невеликою кількістю ітераційних кроків) навчають мережу не тільки для функції XOR, а й для усіх булевих функцій двох змінних.

OPERATOR GENETIC ALGORITHM AND TRAINING OF NEURAL NETWORK

Oliinyk L. O.
Cand. Sc. (Phys.-Math.), PhD,
Associate Professor, Director
Separate Structural Unit Technological Professional College
of Dniprovsky State Technical University
Konstytutsii Ave., 2A, Kamianske, Dnipropetrovsk region, Ukraine
orcid.org/0000-0002-4392-0048
l.olejnik57@gmail.com

Key words: *involutive operator, stochastic operator, crossover, mutation, fitness function, n-dimensional hypercube, binary code, Gray code.*

This work is dedicated to the study of the effectiveness of the application of the operator genetic algorithm for the training of neural networks.
As is known, genetic algorithms, their various interpretations, are quite effective tools for finding solutions to optimization problems. The action of the genetic algorithm is based on the random process of forming a population of possible solutions, among which the best in a certain sense are selected.
The article presents the operator model of the genetic algorithm (then – the operator genetic algorithm) developed by the author, which is used for neural network training. The operator genetic algorithm is based on the application of involutive operators acting in the Cartesian product of two instances of n-dimensional Euclidean space and carrying out crossover and mutation operations. In addition, stochastic operators operating in the same Cartesian product are the main tool for forming populations of chromosome vectors. The result of these operators can be interpreted as generalized crossover and mutation. This operator model makes it possible to form populations of chromosome vectors of small power in the search area at each iterative step of the algorithm. The operator model of the genetic algorithm involves binary coding of involutive operators that transform the vertices of the hypercube, which is the search area, but the points of the search area are represented in decimal format.
The research article examines the effectiveness of the operator algorithm for training a known neural network that implements the Boolean XOR function. The point of consideration of this example is that it is a partial case of the problem of classification of points of a unit hypercube of arbitrary dimension. As a result of the application of the operator genetic algorithm to the training of the XOR neural network, a set of operators is obtained, which effectively (with a small number of iteration steps) trains the network not only for the XOR function, but also for all Boolean functions of two variables.

Вступ

Генетичні алгоритми, як різновид еволюційних алгоритмів, сьогодні мають досить велику популярність при застосуванні до прикладних оптимізаційних задач. Тому розробка алгоритмів, які б удосконалювали або узагальнювали теоретичні основи генетичного алгоритму є актуальною проблемою в теорії еволюційних алгоритмів.

В цій роботі наведено теоретичні засади операторної моделі генетичного алгоритму, яка є новим підходом до розв'язання задач оптимізації, що базується на застосуванні лінійних інволютивних і стохастичних операторів, що діють у декартовому добутку двох екземплярів n-вимірного евклідового простору. Інволютивні оператори виконують перетворення вершин гіперкубу, який є областю пошуку розв'язків. Стохастичні оператори виконують перетворення області пошуку і формують популяції векторів-хромосом на кожному кроці роботи алгоритму, що дозволяє розглядати популяції векторів-хромосом невеликої потужності. Дія цих операторів в частинних випадках є еквівалентною класичному кросоверу і мутації, а в загальному випадку дія операторів є узагальненням кросоверу та мутації. Тому автор вважає можливим користуватись назвою «операторний генетичний алгоритм».

Принциповою відмінністю даного алгоритму від класичного генетичного алгоритму є те, що операторний алгоритм не потребує дискретизації

області пошуку (тобто покриття рівномірною сіткою). Початкова популяція формується з множини вершин *n*-вимірного гіперкубу і на кожному ітераційному кроці послідовність наближень розв'язку визначається вершинами новоутворених областей пошуку («вкладених» гіперкубів). Крім того необхідно зауважити, що точки області пошуку представляються у десятковому форматі, а інволютивним операторам, які виконують перетворення вершин гіперкубу, ставляться у відповідність бінарні коди.

Збіжність послідовності наближень розв'язку задачі оптимізації зумовлюється тим, що лінійні оператори, які використовуються, є унітарними або стискаючими. Операторний генетичний алгоритм апробовано на задачах пошуку глобального мінімуму у заданій області ([1; 2; 3]).

В цій роботі операторний генетичний алгоритм застосовано до навчання відомої нейронної мережі, що реалізує булеву функцію XOR. Цей приклад є важливим, тому що представляє частинний випадок задачі класифікації точок одиничного гіперкуба довільної вимірності. Результати роботи операторного алгоритму порівняно з результатами застосування класичного генетичного алгоритму наведеними в роботі ([5]).

Актуальність даного дослідження полягає у розробці математично обґрунтованого алгоритму, що на теперішній час є важливим аспектом розробки теоретичних основ еволюційних алгоритмі. В роботі ([4]), відмічається, що: «Особливо проблемним є напрям розробки метаевристичних алгоритмів, де в гонитві за науковою новизною автори створюють все нові алгоритми, не даючи їхнього математичного обґрунтування або більшменш вичерпного експериментального аналізу і порівняння з вже існуючими алгоритмами».

Огляд наявної літератури. З тематики присвяченої генетичному алгоритму і його застосуванням існує велика кількість робіт, серед яких виділимо підручник Глибовця М. М. і Гулаєвої Н. М. «Еволюційні алгоритми» ([6]), у якому на високому науковому рівні викладено теорію генетичного алгоритму і його застосувань. Крім того підручник містить великий обсяг літературних джерел. Що стосується теорії нейронних мереж, то це – робота Саймона Хайкіна «Neural Networks A Comprehensive Foundation» ([7]), де зібрано і викладено максимальний обсяг теоретичного і практичного матеріалу з цієї тематики.

Постановка задачі: розробка, теоретичне обґрунтування і опис операторного генетичного алгоритму, апробація його на відомих прикладних задачах.

Мета роботи: дослідити можливості операторного генетичного алгоритму при застосуванні його до розв'язання задачі навчання нейронних мереж, зокрема нейронної мережі, що реалізує булеву функцію XOR, а також ефективність такого застосування.

Операторний генетичний алгоритм.

Розглянемо задачу пошуку мінімального значення деякої функції $F(x)$ визначеної в замкненій області $\Omega \in R^n$, що визначається наступним чином:

$$\Omega = \left\{ \left(\xi_1, ..., \xi_n \right) : a_m \le \xi_m \le b_m,\ m = \overline{1, n} \right\},$$

1. Формування початкової множини (популяції) розв'язків.

Область Ω є деякий гіперкуб у просторі R^n з «кубічною» нормою $\|x\| = \max\limits_j \left\{ |\xi_j| \right\}$. Вершинам даного гіперкуба $\Omega \in R^n$ однозначно привласнюються бінарні коди Грея, таким чином, що вершина з мінімальними значеннями координат $A = (a_1, ..., a_n)$ має нульовий код, а вершина з максимальними значеннями координат $B = (b_1, ..., b_n)$ має одиничний код. Гіперкуб $\Omega \in R^n$ має 2^n вершин, які складають 2^{n-1} пар «протилежних» вершини, що знаходяться на максимальній відстані Хеммінга.

Нехай $\sigma = (\sigma_1, ..., \sigma_n)$ двійковий вектор (код Грея), позначимо відповідну йому вершину гіперкубу $A_\sigma = \left(w(\sigma_1), ..., w(\sigma_n) \right)$, $w(\sigma_i) = \begin{cases} a_i, & \sigma_i = 0 \\ b_i, & \sigma_i = 1 \end{cases}$, тоді «протилежна» вершина має вигляд $A_{\bar{\sigma}} = \left(w(\bar{\sigma}_1), ..., w(\bar{\sigma}_n) \right)$, де $\bar{\sigma} = (\bar{\sigma}_1, ..., \bar{\sigma}_n)$. Зрозуміло, що відстань Хеммінга між бінарними векторами $\sigma = (\sigma_1, ..., \sigma_n)$ і $\bar{\sigma} = (\bar{\sigma}_1, ..., \bar{\sigma}_n)$ є максимальною і дорівнює $\chi(\sigma, \bar{\sigma}) = n$.

Для визначення координат усіх вершин гіперкуба, достатньо мати координати двох вершин $A = (a_1, ..., a_n)$ та $B = (b_1, ..., b_n)$. Для операторної матриці

$$\hat{P}_j^k = \begin{pmatrix} P_j^k & I - P_j^k \\ I - P_j^k & P_j^k \end{pmatrix},$$

де $P_j^k x = P_j^k \left(\sum\limits_{i=1}^{n} \xi_i \mathrm{e}_i \right) = \sum\limits_{i=j}^{k} \xi_i \mathrm{e}_i$,

де $\forall x = (\xi_1, ..., \xi_n) \in R^n$ і $\{e_1, ..., e_n\} \subset R^n$ ортонормований базис, приймемо позначення $\hat{P}(\sigma)$, де σ – двійковий код, що визначає оператор переставлення координат вершин гіперкубу (оператор кросоверу). Застосовуючи оператори $\hat{P}(\sigma)$ до вектора, складеного з вершин $A = (a_1, ..., a_n)$ та $B = (b_1, ..., b_n)$, отримаємо усі максимально віддалені пари вершин.

Кількість операторів $\hat{P}(\sigma)$ залежить від вимірності простору R^n, а саме, дорівнює $2^{n-1} - 1$.

Саме вершини гіперкуба, тобто граничні точки області Ω, утворюють початкову множину наближень розв'язку задачі (початкову популяцію векторів-хромосом). Зрозуміло, що при великих значеннях *n* ця процедура потребує великих обсягів обчислення. Тому можна обмежуватись випадковим вибором певної невеликої кількості операторів $\hat{P}(\sigma)$ та, відповідно, відбирати деяку множину вершин гіперкубу.

1. *Побудова нової області пошуку (першої популяції)*

Для пошуку мінімального значення функції $F(x)$ обираємо 2^{n-1} пар вершин гіперкубу, відстань Хеммінга між кодами Грея яких є максимальною $\chi(\sigma,\bar{\sigma}) = n$. Отже, це пари вершин гіперкубу:

$$A_{\sigma_j} = W(\sigma_j) = \left(w(\sigma_{j1}),...,w(\sigma_{jn})\right)$$

$$A_{\bar{\sigma}_j} = W(\bar{\sigma}_j) = \left(w(\bar{\sigma}_{j1}),...,w(\bar{\sigma}_{jn})\right),\ j = 0,...,2^{n-1}.$$

Розглянемо вектори

$$\hat{X}_j = \begin{pmatrix} W(\sigma_j) \\ W(\bar{\sigma}_j) \end{pmatrix} = P(\sigma)\begin{pmatrix} A \\ B \end{pmatrix} \in R^n \times R^n,\ j = 0,...,2^{n-1}.$$

Діючи на ці вектори операторами

$$\hat{P}(\alpha),\hat{Q}(\beta) \in L\left(R^n \times R^n; R^n \times R^n\right)$$

$$\hat{P}(\alpha) = \begin{pmatrix} P(\alpha) & I - P(\alpha) \\ I - P(\alpha) & P(\alpha) \end{pmatrix},\ \hat{Q}(\beta) = \begin{pmatrix} I - Q(\beta) & Q(\beta) \\ Q(\beta) & I - Q(\beta) \end{pmatrix},$$

$$\hat{P}(\alpha)\hat{X}_j = \hat{P}(\alpha)P(\sigma)\begin{pmatrix} A \\ B \end{pmatrix} = \hat{Z}_j = \begin{pmatrix} z_{j1} \\ z_{j2} \end{pmatrix},$$

$$\hat{Q}(\beta)\hat{X}_j = \hat{Q}(\beta)P(\sigma)\begin{pmatrix} A \\ B \end{pmatrix} = \hat{V}_j = \begin{pmatrix} v_{j1} \\ v_{j2} \end{pmatrix}$$

де $P(\alpha), Q(\beta) \in L\left(R^n; R^n\right)$ які діють наступним чином: $\forall x = (\xi_1,...,\xi_n) \in R^n$

$$P(\alpha)x = P(\alpha)\left(\sum_{i=1}^{n} \xi_i e_i\right) = \sum_{i=1}^{n} \alpha_i \xi_i e_i,$$

$$Q(\beta)x = Q(\beta)\left(\sum_{i=1}^{n} \xi_i e_i\right) = \sum_{i=1}^{n} \beta_i \xi_{n-i+1} e_i,$$

де $\alpha = (\alpha_1,...,\alpha_n), \beta = (\beta_1,...,\beta_n) \in R^n$, $\{e_1,...,e_n\} \subset R^n$ базис, отримаємо:

$$\hat{P}(\alpha) = \begin{pmatrix} P(\alpha) & I - P(\alpha) \\ I - P(\alpha) & P(\alpha) \end{pmatrix},\ \hat{Q}(\beta) =,$$

$$= \begin{pmatrix} I - Q(\beta) & Q(\beta) \\ Q(\beta) & I - Q(\beta) \end{pmatrix},$$

де $\alpha = (\alpha_1,...,\alpha_n)$, $\beta = (\beta_1,...,\beta_n)$, $0 \le \alpha_i < 1$, $0 \le \beta_i < 1$, стохастичні вектори.

В результаті отримаємо вектори, компоненти яких є точками області пошуку, для яких обчислюються значення функції $F(x)$.

$$F(z_{jk}),\quad F(v_{jk}),\quad k = 1,2,\quad j = 1,...,2^n.$$

Позначимо z^0 точку області пошуку, для якої функція має найменше з отриманих значень, тобто:

$$F(z^0) = \min_{j,k}\left\{F(z_{jk}), F(v_{jk})\right\},\quad k = 1,2,\quad j = 1,...,2^n$$

Точка z^0 розбиває гіперкуб Ω на 2^n гіперкубів, для яких вона буде однією з вершин. Протилежними до неї вершинами будуть вершини $A_{\sigma_j} = W(\sigma_j) = \left(w(\sigma_{j1}),...,w(\sigma_{jn})\right)$ початкового гіперкубу Ω.

Нехай A_j – деяка фіксована вершина початкового гіперкубу. Розглянемо гіперкуб, де A_j та z^0 пара максимально віддалених вершин. Позначимо цей гіперкуб наступним чином: $\Omega(A_j)$. Якщо $\sigma_j = (\sigma_{j1},...,\sigma_{jn})$ двійковий код вершини A_j, то бінарний код вершини z^0 має вигляд

$$\bar{\sigma}_j = (\bar{\sigma}_{j1},...,\bar{\sigma}_{jn}).$$

Розглянемо вектор $\hat{X}_j = \begin{pmatrix} W(\sigma_j) \\ z^0 \end{pmatrix} = \begin{pmatrix} A_j \\ z^0 \end{pmatrix}$.

Застосовуючи оператори $P(\sigma_j)$ до вектора $\hat{X}_j$, отримаємо усі вершини гіперкубу $\Omega(A_j)$. До кожного вектору множини пар максимально віддалених вершин гіперкубу $\Omega(A_j)$ застосуємо оператори $\hat{P}(\alpha),\hat{Q}(\beta)$. В результаті отримаємо точки (вектори) всередині гіперкубу $\Omega(A_j)$, для яких обчислюються значення функції $F(x)$.

Позначимо z_j^0 точку гіперкубу $\Omega(A_j)$, для якої функція має найменше з отриманих значень, тобто:

$$F(z_j^0) = \min_{j,k}\left\{F(z_{jk}), F(v_{jk})\right\},\quad k = 1,2,\quad j = 1,...,2^n$$

Переглянувши послідовно усі вершини початкового гіперкубу A_j, гіперкуби $\Omega(A_j)$, отримаємо сукупність точок z_j^0, $j = 1,...,2^n$, для яких функція має найменше значення у відповідних гіперкубах $\Omega(A_j)$. Позначимо точку z^1. таку, що $F(z^1) = \min_j\left\{F(z_j^0)\right\}$.

Розглянемо пару точок z^0 та z^1. Будуємо вектор $\begin{pmatrix} z^0 \\ z^1 \end{pmatrix}$. Застосовуючи до нього оператор $\hat{P}(\alpha)$ при $\alpha = (2,...,2)$, отримаємо

$$\hat{P}(\alpha)\begin{pmatrix} z^0 \\ z^1 \end{pmatrix} = \begin{pmatrix} 2z^0 - z^1 \\ 2z^1 - z^0 \end{pmatrix}.$$

Якщо $F(z^0) < F(z^1)$, то розглянемо пару точок $A_1 = 2z^0 - z^1$ і $B_1 = z^1$. Для цієї пари точок має місце рівність $\|A_1 - z^0\| = \|B_1 - z^0\|$. Якщо $F(z^0) > F(z^1)$, то розглянемо пару точок $A_1 = 2z^1 - z^0$ і $B_1 = z^0$. Для цієї пари точок має місце рівність $\|A_1 - z^1\| = \|B_1 - z^1\|$.

Отримані точки A_1, B_1 є максимально віддаленими вершинами нового гіперкубу Ω_1 з центром в точці z_1, яка дорівнює z^0 або z^1 в залежності від виконання умов $F(z^0) < F(z^1)$ або $F(z^0) > F(z^1)$. Центром гіперкубу є точка z_1 з найкращим на даному кроці екстремальним значенням функції $F(z_1)$. z_1 – є першим наближенням розв'язку задачі.

Вершини гіперкубу Ω_1 отримаємо за відомою вже процедурою $\hat{P}(\sigma)\begin{pmatrix} A_1 \\ B_1 \end{pmatrix}$, де σ перебігає множину кодів операторів перестановок координат. Отримані таким чином вершини гіперкубу Ω_1 утворюють нову множину область пошуку або першу популяцію векторів-потомків.

Наступний ітераційний крок пошуку наближеного розв'язку полягає у повторенні описаної вище процедури застосованої до гіперкубу Ω_1.

В результаті отримаємо новий гіперкуб Ω_2 з центром в точці z_2. Якщо виконується нерівність $F(z_1) > F(z_2)$, то переходимо до наступного ітераційного кроку.

Так, за умови постійного спадання значень функції у кожному новому наближенні розв'язку, продовжуємо ітераційний процес і отримаємо на деякому кроці послідовність гіперкубів Ω_m і наближень $z_1, z_2, ..., z_m$ таку, що $F(z_1) > F(z_2) > ... > F(z_m)$. Послідовність $z_1, z_2, ..., z_m$ прямує до розв'зку задачі із зростанням номеру m.

Критерій зупинки алгоритму визначається точністю обчислень або величиною похибки ε. Якщо відоме мінімальне значення функції, то розглядається виконання нерівності $|F(z_m) - F(z_{\min})| < \varepsilon$. Якщо ж мінімальне значення функції невідоме, то розглядається різниця між двома послідовними наближеннями розв'язку $\|z_{m-1} - z_m\| < \varepsilon$.

2. Мутація

У зв'язку з тим, що ітераційний процес має стохастичний характер, монотонність послідовності значень функції $F(z_1) > F(z_2) > ... > F(z_m)$ може порушуватись. На деякому ітераційному кроці можуть виникнути наступні ситуації:

1. $F(z_{k-1}) \le F(z_k)$.

2. Гіперкуб Ω_k вироджується у многовид меншої вимірності (гіперплощина, точка, відрізок), тобто не буде опуклою множиною). В цьому випадку необхідно в такий ітераційний крок вводити додаткову дію аналогічну операції мутації для класичного генетичного алгоритму.

а) Узагальнена мутація.

Якщо виникає ситуація 1 і при цьому гіперкуб Ω_k є опуклою областю в R^n, то необхідно дослідити функцію в околі гіперкубу Ω_k, тобто обчислити значення функції в точках деякої множини, яка містить в собі Ω_k. Для цього пропонується процедура аналогічна процесу «ігор хаосу». А саме, застосовуючи оператор $\hat{P}(\alpha)$ при $\alpha = (2, ..., 2)$ до вектора $\begin{pmatrix} A_k \\ B_k \end{pmatrix}$ розтягнемо вектор в три рази, а потім, (як це показано вище) після дії операторів отримаємо вершини $\hat{P}(\sigma)\hat{P}(\alpha)\begin{pmatrix} A_k \\ B_k \end{pmatrix}$ розтягнутого гіперкубу $\tilde{\Omega}_k \supset \Omega_k$. Вершини гіперкубу $\tilde{\Omega}_k$ є атракторами. Вибравши точку z_k і випадковим чином обираємо один з атракторів, який позначимо $\tilde{A}_k$. Розглянемо вектор $\begin{pmatrix} z_k \\ \tilde{A}_k \end{pmatrix}$. Подіємо на цей вектор оператором $\hat{P}(\alpha)$, де $\alpha = \left(\frac{1}{3}, ..., \frac{1}{3}\right)$, отримаємо точку M_1, яка не належить гіперкубу Ω_k:

$$\hat{P}(\alpha)\begin{pmatrix} z_k \\ \tilde{A}_k \end{pmatrix} = \left(\frac{1}{3}z_k + \frac{2}{3}\tilde{A}_k\right) = M_1.$$

Аналогічно, обираючи випадковим чином атрактор $\tilde{A}_k$, отримаємо точку

$$M_2 = \hat{P}(\alpha)\begin{pmatrix} M_1 \\ \tilde{A}_k \end{pmatrix} = \left(\frac{1}{3}M_1 + \frac{2}{3}\tilde{A}_k\right).$$

Продовжуючи цей випадковий процес, отримаємо певну кількість точок, що не належать гіперкубу Ω_k. Кількість таких точок m визначається в залежності від вимірності простору, в якому розв'язується задача, і складності функції $F(x)$. Цей процес можна запускати кілька разів, доки не з'явиться точка M^* така, що $F(z_{k-1}) > F(M^*)$. Після чого ітераційний пошук розв'язку продовжується згідно п.2.

б) Якщо виникає ситуація 2, то необхідно повернутись до попереднього ітераційного кроку і застосувати процедуру узагальненої мутації.

Працездатність та ефективність представленого алгоритму розглянуто на прикладі застосування до навчання нейронної мережі, що реалізує булеву функцію XOR (або логічне додавання $x \oplus y$). Результат роботи алгоритму порівнюються з наведеними даними в роботі ([5]).

Навчання нейронної мережі, що реалізує булеву функцію XOR.

Як відомо ([5],[7]), функцію XOR реалізує двошарова нейронна мережа вигляду (рис. 1), де x_{1j}, x_{2j} вхідні сигнали (образи, дані).

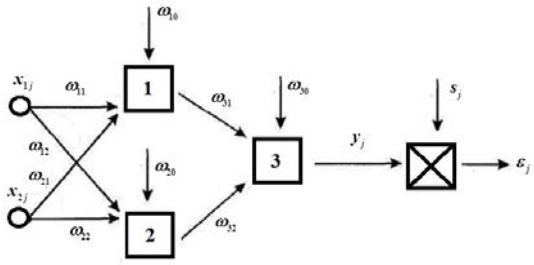

Рис. 1.

Навчання нейронної мережі полягає у мінімізації функції похибки, яка має вигляд ([5]):

$$\varepsilon = \frac{1}{4}\sum_{j=1}^{4}\varepsilon_j^2 = \frac{1}{4}\sum_{j=1}^{4}\left(s_j - y_j\right)^2 \to \min$$

$$y_j = \frac{\omega_{00}}{1 + \exp\left(\frac{-\omega_{31}}{1 + \exp\left(-\omega_{11}x_{1j} - \omega_{12}x_{2j} - \omega_{10}\right)} + \frac{-\omega_{32}}{1 + \exp\left(-\omega_{21}x_{1j} - \omega_{22}x_{2j} - \omega_{20}\right)} - \omega_{30}\right)}$$

$$-100 \le \omega_{kj} \le 100, \quad k = 1, 2, 3, \quad j = 0, 1, 2, \quad \omega_{00} = 1 \qquad (1)$$

x_{kj}, $k = 1, 2$, $j = 1, 2, 3, 4$ – значення вхідних сигналів (образів, даних)

Функція ε похибки, є функцією десятьох змінних, отже маємо задачу пошуку мінімального значення функції

$$\varepsilon\left(\omega_{10}, \omega_{20}, \omega_{30}, \omega_{11}, \omega_{12}, \omega_{21}, \omega_{22}, \omega_{31}, \omega_{32}, \omega_{00}\right)$$

визначеної в замкненій області $\Omega \in R^{10}$, що визначається нерівностями (1).

Так як n=10, то область пошуку обмежено десятивимірним гіперкубом, який має 1024 вершин (512 пар протилежних максимально віддалених вершин). Процедура перебору усіх пар вершин потребує великих обсягів обчислення. Тому обмежимось випадковим вибором 25 (2,44%) операторів перебору пар вершин $\hat{P}(\sigma)$ які відбиратимуть деяку множину, що складається з 50 (4,88%) вершин гіперкубу.

Для навчання даної мережі визначено 25 операторів $\hat{P}(\sigma)$ з десятковими кодами (табл. 1).

Вектори α і β визначаються випадковим чином. Множина операторів також визначається випадково (в цьому експерименті 25 операторів, що надали позитивний результат, визначились за 4 спроби). При цих даних навчання нейронної мережі дає позитивний результат для усіх булевих функцій двох змінних.

У таблиці 2 наведено результати навчання нейронної мережі для відомих булевих функцій (F) двох змінних, а саме, значення параметрів нейронної мережі. k – кількість ітераційних кроків операторного генетичного алгоритму.

У таблиці 3 наведено результати роботи нейронної мережі у порівнянні із значеннями відомих булевих функцій.

В роботі ([5]), показано, що класичний алгоритм при навчанні нейронної мережі, що реалізує булеву функцію XOR, на 75 поколінні досяг значення похибки $7 \cdot 10^{-5}$. Розмірність популяції дорівнює 77, довжина хромосом 99 бітів. Область пошуку параметрів [-10; 10]. В наведеному прикладі класичний алгоритм застосовувався з використанням програмного засобу FlexTool.

Як видно з таблиць 2, 3 результати операторного алгоритму значно кращі, для похибки $4,78 \cdot 10^{-8}$, при цьому виконується всього 10 ітераційних кроків, тобто 10 поколінь. На кожному кроці роботи алгоритму використовуються 25 інволютивних операторів, які формують множину векторів-хромосом, що складається з 50 вершин гіперкубів. При цьому область пошуку параметрів [-100;100] значно більша за обсягом. Нарешті, визначена множина інволютивних операторів (таблиця 1) і стохастичних операторів з великим ступенем точності для похибки, визначає параметри нейронної мережі для усіх булевих функцій від двох змінних (таблиці 1, 2).

У таблиці 4 наведено результати навчання нейронної мережі для відомих булевих функцій (F) двох змінних для стохастичних операторів $\hat{P}(\alpha), \hat{Q}(\beta)$, що визначені вище і 50-ти інволютивних операторів, на відміну 25 операторів наведених у таблиці 1.

Як видно з таблиць 2 і 4, результати роботи алгоритму суттєво не відрізняються, тому можна стверджувати, що кількість вершин гіперкубу можуть прискорювати роботу алгоритму, але величина

Таблиця 1

Коди інволютивних операторів

№	1	2	3	4	5	6	7	8	9	10	11	12	
Код	451	951	992	994	794	79	269	902	569	289	5	878	
№	13	14	15	16	17	18	19	20	21	22	23	24	25
Код	708	613	181	993	645	535	857	236	157	643	816	38	729

Таблиця 2

Параметри нейронної мережі і значення функції похибки

k	F	ω_{11}	ω_{12}	ω_{21}	ω_{22}	ω_{31}	ω_{32}	ω_{10}	ω_{20}	ω_{30}	ε
6	$\wedge$	8,865	12,811	16,867	19,120	17,913	19,493	-18,418	-20,89	-16,367	5,11E-13
10	$\oplus$	-12,137	23,050	14,505	-19,590	18,793	17,155	-20,955	-8,957	-8,548	4,78E-08
3	$\vee$	-17,519	34,120	-43,053	-43,529	-30,196	-36,279	30,273	7,740	43,498	2,09E-12
5	$\Leftrightarrow$	-8,153	-31,161	20,560	13,863	37,061	20,806	1,260	-33,302	-5,858	4,21E-06
6	$\rightarrow$	-12,825	2,019	-31,838	34,554	8,191	31,465	20,472	3,469	-23,514	3,32E-14

Таблиця 3

Порівняння результатів роботи нейронної мережі із значеннями відомих булевих функцій

Вхідні сигнали		**Значення булевих функцій**				
x1	x2	$\wedge$	$\oplus$	$\vee$	$\Leftrightarrow$	$\rightarrow$
0	0	0	0	0	1	1
0	1	0	1	1	0	1
1	0	0	1	1	0	0
1	1	1	0	1	1	1
Результати роботи нейронної мережі		7,8E-08	0,000194	1,07E-10	1	1
		1,42E-06	0,999722	0,999998	0,002848	1
		1,1E-07	0,999805	0,999998	0,002956	2,21E-07
		1	0,000194	0,999998	0,999946	1

Таблиця 4

Параметри нейронної мережі і значення функції похибки

k	F	ω_{11}	ω_{12}	ω_{21}	ω_{22}	ω_{31}	ω_{32}	ω_{10}	ω_{20}	ω_{30}	ε
4	∧	7,880	34,764	-43,360	-30,741	36,354	30,834	-37,489	-30,59	-14,986	2,15E-12
6	⊕	-16,410	40,794	30,600	-38,770	36,380	27,471	-36,827	-26,25	-6,130	2,35E-06
22	∨	9,341	9,129	-10,521	-10,005	10,185	-10,776	-6,596	6,569	1,729	3,83E-09
5	⇔	16,410	46,100	-35,75	-38,77	36,380	31,465	-53,11	16,477	-23,15	1,5E-08
4	→	13,183	43,815	-32,82	32,751	7,657	23,053	12,173	15,865	-15,68	2,6E-08

похибки при цьому відрізняється незначно. Отже, для розв'язання задачі навчання мережі можна обмежитись меншою кількістю інволютивних операторів, що зменшує кількість обчислень.

Висновки

Застосування операторного генетичного алгоритму до задачі навчання нейронної мережі, що реалізує булеву функцію XOR, показало працездатність представленого алгоритму і ефективність його роботи. В результаті застосування операторного генетичного алгоритму отримано множину операторів, які ефективно (з невеликою кількістю ітераційних кроків) визначають параметри нейронної мережі, тобто навчають її не тільки для булевої функції XOR, а й для усіх можливих булевих функцій двох змінних.

Операторний генетичний алгоритм представляє досить ефективний швидкодіючий інструмент розв'язання задач оптимізації. Важливими питаннями, які потребують подальших досліджень є:

– визначення області застосування даного алгоритму;

– визначення працездатності для просторів великої вимірності;

– визначення можливості управління добором операторів, що задіяні в роботі алгоритму і формують область пошуку розв'язків.

ЛІТЕРАТУРА

1. Л.О. Олійник Операторна модель рекомбінації в генетичних алгоритмах. Математичне моделювання. 2019. Вип. 1(40). Кам'янське. ДДТУ. С. 14–21.
2. Л.О. Олійник, Д.Л. Олійник Про ефективність операторної модифікації генетичного алгоритму в задачах двовимірної оптимізації. МНЖ «Грааль науки» № 11 (грудень 2021). С. 221–229.
3. Л.О. Олійник, О.О. Довженко Демонстраційний програмний засіб тривимірної операторної моделі генетичного алгоритму. «Математичні проблеми технічної механіки – 2023», Міжнародна наукова конференція, том 2, тези доповіді, Київ, Дніпро, Кам'янське. С. 5–7.
4. Н.М. Гулаєва, В.П. Шило, М.М. Глибовець Генетичні алгоритми як обчислювальні методи скінченновимірної оптимізації. Cybernetics and Computer Technologies. 2021, № 3, С. 5–14.
5. Кононюк А.Ю. Нейроні мережі і генетичні алгоритми – К.: «Корнійчук», 2008. 446 с.
6. М.М. Глибовець. Н.М. Гулаєва Еволюційні алгоритми: підручник. – К.: НаУКМА, 2013. 828 с.
7. Simon Haykin Neural Networks A Comprehensive Foundation, Second edition. – Prentice Hall, New Jersey. 2016. 1104 p.

REFERENCES

1. L. Oliinyk (2019) Operatorna model recombinacii v genetichnish algoritmash (Operator model of recombination in genetic algorithms). Mathematical modeling. – Issue 1(40). Kamianske. DDTU (in Ukrainian). P. 14–21.
2. L. Oliinyk, D. Oliinyk (2021) Pro efectivnist operatornoi modificacii genetichnogo algoritmu v zadachash dvovimirnoi optimizacii (On the effectiveness of operator modification of the genetic algorithm in two-dimensional optimization problems). – "Grail of Science" magazine № 11. (in Ukrainian). P. 221–229.
3. L.Oliynyk, O.Dovzhenko (2023) Demonstraciynyi programnyi zasib tryvymirnoi operatornoi modeli genetichnogo algoritmu (Demonstration software of the three-dimensional operator model of the genetic algorithm). "Mathematical problems of technical mechanics – 2023", International scientific conference, volume 2, theses of the report, Kyiv, Dnipro, Kamianske. (in Ukrainian). P. 5–7.
4. N. Gulaeva, V. Shilo, M. Hlybovets. Genetichni algorytmi yak obchysluvalni metody skinchennovymirnoii optymizacii. (Genetic Algorithms as Computational Methods for Finite-Dimensional Optimization). Cybernetics and Computer Technologies. 2021, № 3, c. 5–14.
5. Kononiuk A. Nejronni meregi I genetichni algorytmi (Neural networks and genetic algorithms). K.:"Korniichuk", 2008. 446 p.
6. M. Hlybovets. N.Gulaeva (2013) Evolyuciyni algorytmi (Evolutionary algorithms): textbook.-K.: NaUKMA, (in Ukrainian). 828 p.
7. Simon Haykin Neural Networks A Comprehensive Foundation, Second edition. – Prentice Hall, New Jersey. 2016. 1104 p.