УДК 681.51

В.А. Жеребко

ПРОГРАМНА МОДЕЛЬ ГЕНЕТИЧНОГО ОПТИМІЗАТОРА ВАГ ШТУЧНОЇ НЕЙРОННОЇ МЕРЕЖІ В LABVIEW

У даній роботі представлена процедура моделювання та синтезу штучної нейронної мережі із використанням генетичного алгоритму. Як приклад штучної нейронної мережі обрано штучну нейронну мережу, що реалізує логічну функцію «XOR» (виключаюче АЛЕ). Синтез та моделювання виконано із використанням програмного забезпечення LabVIEW. Структура XOR-нейронної мережі – двошаровий персептрон із одним прихованим шаром нейронів, а навчання XOR-нейронної мережі відбувається класичним генетичним алгоритмом з турнірною селекцією.

Вступ

Проектування надвеликих оптимальних систем управління та керування складними технологічними й виробничими процесами вимагає від системного аналітика та інженерного складу розробників системи використання сучасних інтелектуальних технологій, зокрема математику штучних нейронних мереж (ШНМ) [1].

При вирішенні багатьох оптимізаційних задач актуальною є проблема знаходження глобального оптимуму цільової функції керування в багатовимірному просторі керованих змінних. Традиційні методи багатовимірної оптимізації є методами локального пошуку, що сильно залежать від вибору початкової точки пошуку та накладають додаткові обмеження на властивості цільової функції оптимізації.

Одним із прикладів альтернативних інтелектуальних методів оптимізаційного пошуку є так звані еволюційні підходи, наприклад, генетичні алгоритми (ГА).

Проблематика ШНМ й ГА [2] та комбінація цих методів – це одна з областей досліджень, що інтенсивно розвиваються в даний час. Її можна вважати сучасним відгалуженням інформатики, пов'язаним з методами штучного інтелекту.

ШНМ і ГА доповнюють класичні експертні системи, які вважаються одним з головних напрямів технологій штучного інтелекту, а також в деяких випадках виконують функції цих систем шляхом реалізації так званих інтелектуальних обчислювальних систем (ІОС). Останні є об'єднанням ШНМ та ГА, взаємодія яких дозволяє вирішувати різні завдання, але найважливіше – вони стають універсальним інструментарієм для обробки інформації.

ІОС можна легко «перепрограмувати» на рішення іншої задачі, причому роль такого програмування відіграє процедура навчання. Таким чином, ІОС володіють здібністю до навчання, що вважається головним атрибутом інтелектуальності.

ГА є метод, що відображає природну еволюцію методів вирішення проблем, і насамперед завдань оптимізації. ГА – це процедури пошуку, засновані на механізмах природного відбору і спадкоємства. У них використовується еволюційний принцип виживання найбільш пристосованих особин. Вони відрізняються від традиційних методів оптимізації декількома базовими елементами. Зокрема, ГА:

1) обробляють не значення параметрів завдання, а їх закодовану форму;

2) здійснюють пошук оптимального рішення виходячи не з єдиної початкової точки, а з їх деякої популяції;

3) використовують тільки цільову санкцію, а не її похідні або іншу додаткову інформацію;

4) застосовують стохастичні, а не детерміновані правила вибору.

Перераховані чотири властивості, які можна сформулювати також як кодування параметрів, операції над популяціями, використання мінімуму інформації про завдання і рандомізація операцій призводять в результаті до стійкості ГА і до їх переваги над іншими широко вживаними інтелектуальними технологіями оптимізаційного пошуку.

В даній роботі розглядається приклад простої ШНМ із архітектурою персептрона, процедура навчання якої виконується із використанням ГА, а реалізацію виконано засобами пакету LabVIEW.

1. Постановка задачі

Кожен нейрон ШНМ типу персептрон [3] є формальним пороговим елементом, що набуває одиничних значень у випадку, якщо сумарний зважений вхід більше деякого порогового значення Θ:

$$Y_j = \begin{cases} 0, & \sum_{i=1}^{n} X_i \cdot w_i < \Theta \\ 1, & \sum_{i=1}^{n} X_i \cdot w_i \geq \Theta \end{cases}.$$

Таким чином, при заданих значеннях вагових коефіцієнтів w_i і порогів Θ_j, нейрон має певне вихідне значення для кожного можливого вектора входів. Множина вхідних векторів, за яких нейрон активний (Y_j=1), відокремлена від множини векторів, за яких нейрон пасивний (Y_j=0), гіперплощиною, рівняння якої

$$\sum_{i=1}^{n} X_i \cdot w_i = \Theta.$$

Отже, нейрон здатний відокремити тільки такі дві множини векторів входів, для яких існує гіперплощина, що відсікає одну множину від іншої. Такі множини називають лінійно роздільними. Проілюструємо це поняття на прикладі [3].

Розглянемо одношаровий персептрон, що складається з одного нейрона з двома входами. Вхідний вектор містить тільки дві бульові компоненти x_1 і x_2, що визначають площину.

На даній площині можливі значення вхідних векторів відповідають вершинам одиничного квадрата. У кожній вершині задамо потрібне значення виходу нейрона: 1 (біла точка) або 0 (чорна точка).

Потрібно визначити, чи існує такий набір ваг і порогів нейрона, при якому нейрон зможе набути цих значень виходів? На рис. 1 показана одна з ситуацій, коли цього зробити не можливо внаслідок лінійної нероздільності множини білих і чорних точок.

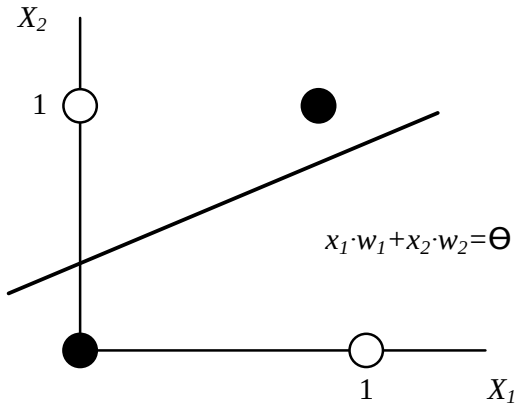

Рис. 1. Білі точки не можуть бути відокремлені однією прямою від чорних

Необхідні виходи нейрона для рис. 1 визначаються табл. 1, в якій легко впізнати завдання логічної функції «виключаюче АБО» (XOR).

Таблиця 1. Логічна функція XOR

X_1	X_2	Y
0	0	0
1	0	1
0	1	1
1	1	0

Неможливість реалізації одношаровим персептроном цієї функції отримала назву *проблеми виключаючого АБО*. Видно, що одношаровий персептрон вкрай унеможливлює точно представити наперед задану логічну функцію XOR.

Хоча даний приклад наочний, він не є серйозним обмеженням для ШНМ. Функція XOR легко реалізується простою двошаровою ШНМ, причому багатьма способами. Один з прикладів такої ШНМ показаний на рис. 2.

Вагові коефіцієнти w_{11}, w_{12}, w_{21}, w_{22} першого шару всі дорівнюють одиниці, вагові коефіцієнти другого шару v_1, v_2 відповідно дорівнюють 1 і -1, а порогові значення Θ усіх нейронів вказані на рис. 2.

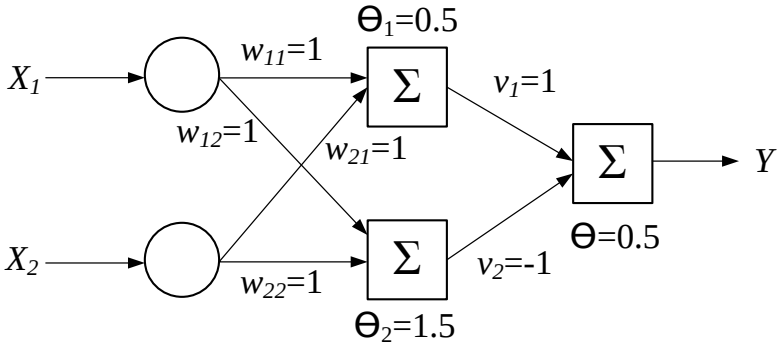

Рис. 2. Структура двошарової ШНМ, що реалізує функцію XOR

Табл. 2. Таблиця істинності для такої ШНМ має вигляд

X_1	X_2	s_1	s_2	y_1	y_2	S	Y
0	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
0	1	1	1	1	0	1	1
1	1	2	2	1	1	0	0

Тут $s_1 = x_1 \cdot w_{11} + x_2 \cdot w_{21}$ – значення, що поступає на вхід першого нейрона першого шару, $s_2 = x_1 \cdot w_{12} + x_2 \cdot w_{22}$ – вхід другого нейрона першого шару; y_1, y_2 – виходи відповідних нейронів першого шару; S – вхідне значення нейрона другого шару; Y – вихід ШНМ.

2. Генетичні алгоритми у штучних нейронних мережах

Об'єднання ГА і ШНМ відоме в літературі під абревіатурою COGANN *(Combinations of Genetic Algorithms and Neural Networks)* [4]. Думка про те, що ШНМ можуть навчатися за допомогою ГА, висловлювалася різними дослідниками. Перші роботи на цю тему стосувалися застосування ГА як методу навчання невеликих однонаправлених ШНМ, але в подальшому було реалізовано застосування ГА для ШНМ більшої розмірності.

Як правило, завдання полягає в оптимізації ваг ШНМ [5], що має апріорі задану топологію (див. рис. 2). Ваги кодуються у вигляді двійкових послідовностей (хромосом). Кожна особина популяції характеризується повним набором ваг ШНМ. Оцінка пристосованості особин визначається функцією пристосованості (*fitness*-функція), що задається у вигляді суми квадратів погрішностей, тобто різниць між очікуваними (еталонними) і фактично отримуваними значеннями на виході ШНМ для різних вхідних даних.

Приведемо два найважливіші аргументи на користь застосування ГА для оптимізації ваг ШНМ. Перш за все, ГА забезпечують глобальний перегляд простору ваг і дозволяють уникати локальні мінімуми. Крім того, ГА можуть використовуватися в завданнях, для яких інформацію про градієнти отримати дуже складно або вона виявляється дуже дорогою.

Еволюційний підхід до навчання ШНМ складається з двох основних етапів. Перший з них – це вибір відповідної схеми представлення ваг міжнейронних зв'язків. Він полягає в ухваленні рішення – чи можна кодувати ці ваги двійковими послідовностями або потрібна якась інша форма. На другому етапі вже здійснюється сам процес еволюції, заснований на ГА.

Після вибору схеми хромосомного представлення, ГА застосовується до популяції особин (хромосом, що містять закодований набір ваг ШНМ) з реалізацією типового циклу еволюції (ГА), що складається з чотирьох кроків.

Крок 1. Декодування кожної особини (хромосоми) поточного покоління для відновлення набору ваг і конструювання відповідної цьому набору ШНМ з апріорі заданою архітектурою і правилом навчання.

Крок 2. Розрахунок загальної середньоквадратичної погрішності між фактичними і еталонними значеннями на всіх виходах ШНМ при подачі на її входи навчальних образів.

Крок 3. Репродукція особин з вірогідністю, відповідно до пристосованості, або згідно рангу (залежно від способу селекції – наприклад, по методу рулетки або ранговому методу).

Крок 4. Застосування генетичних операторів – таких як схрещування, мутація і/або інверсія для отримання нового покоління особин (ваг ШНМ).

3. Програмна реалізація штучної нейронної мережі в LabVIEW

Одним із сучасних схемних елементів систем керування, що використовується для швидкісного паралельного обрахунку масиву даних та реалізації ШНМ є перепрограмовані мікросхеми ПЛІС (FPGA), програмування яких виконується спеціалізованими засобами мовою VHDL.

Підготовчі процедури моделювання й синтез ШНМ здебільшого виконуються засобами MATLAB, але цей програмний пакет не орієнтований на цільову реалізацію. Тож постає достатньо серйозне та актуальне питання вибору єдиного універсального програмного засобу та апаратної платформи для повного циклу проектування ШНМ для використання в ІОС та системах автоматичного керування.

Пропонується для модельних та прикладних досліджень ШНМ в задачах пошуку та оптимізації законів керування використовувати універсальний програмний засіб LabVIEW компанії National Instruments, а у якості цільової апаратної виконавчої платформи обрати контролер CompactRIO (далі cRIO) [7]. Пакет LabVIEW дозволяє виконати побудову та синтез ШНМ, провести компіляцію в код VHDL та дослідити витрати ресурсів cRIO в мікросхемі FPGA [8].

В даній роботі наведений приклад реалізації типової ШНМ із архітектурою персептрона, що виконує просту логічну функцію XOR на контролері FPGA платформи cRIO (див. рис. 3).

Середовище графічного програмування LabVIEW не має вбудованих засобів для проектування ШНМ. Враховуючи те, що ШНМ складається із сполучених і взаємодіючих між собою штучних нейронів (див. рис. 2), тому їх реалізація мовою G в пакеті LabVIEW не викликає труднощів.

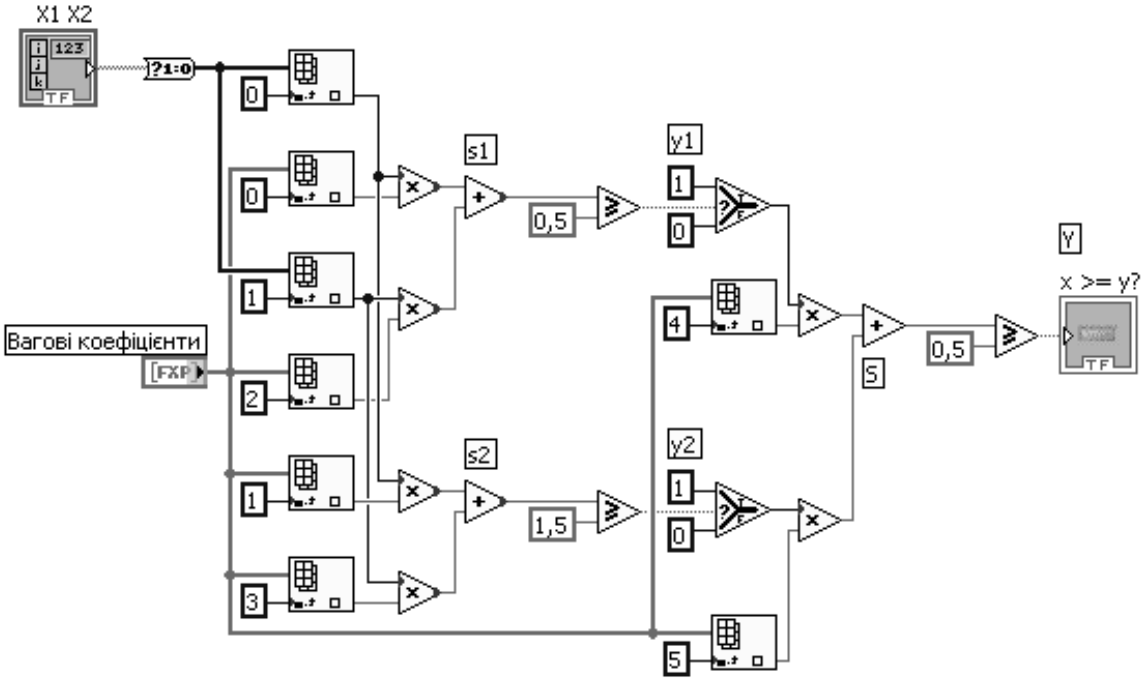

Рис. 3. Блок-діаграма ШНМ, що реалізує функцію XOR в LabVIEW

Для роботи з контролером cRIO необхідно встановити до базового пакету LabVIEW додаткові програмні модулі LabVIEW FPGA та LabVIEW Real-Time. При проектуванні ШНМ та кодуванні їх мовою G на платформі cRIO FPGA потрібно враховувати наступні обмеження засобів LabVIEW FPGA:

– відсутність підтримки багатовимірних (дво- та більше) масивів;

– кількість елементів одновимірних масивів чітко фіксована;

– відсутність типу даних із плаваючою комою (*Floating-Point*);

У LabVIEW FPGA версії 8.5 була введена підтримка формату даних із фіксованою комою (*Fixed-Point*). Операції суми та множення виконуються за допомогою стандартних математичних операцій LabVIEW FPGA. А от операція ділення взагалі відсутня в LabVIEW FPGA. Тому для коректного оперування даними *Fixed-Point* (FXP) доцільно встановити допоміжний модуль LabVIEW FPGA *Fixed-Point Math*.

Масив вагових коефіцієнтів ШНМ складається із шести елементів. ШНМ має два логічні (бітові) входи та один вихід.

Скомпільована блок-діаграма програми зі ШНМ завантажується лише на цільову платформу cRIO FPGA. Інтерфейси всіх підпрограм, за виключенням головної лицьової панелі, вимкнено для економії витрат ресурсів ПЛІС.

В процесі компіляції код G конвертується в код VHDL, а потім у біт-файл конфігурації ПЛІС за допомогою сервера компіляції Xilinx ISE.

ПЛІС cRIO (ядро Virtex-II 3000) має біля 28 тисяч тригерів та більш ніж 3 млн. логічних одиниць (вентилів). Код розробленої програми зі ШНМ був скомпільований на ПК (процесор з подвійним ядром, частота 3ГГц, 2 Гб оперативної пам'яті) за проміжок часу близько 20 хвилин. В результаті об'єм витрачених пар логічних одиниць склав 15 %.

4. Генетичний алгоритм навчання штучної нейронної мережі

Спрощена блок-схема алгоритму, що ілюструє ГА пошукової еволюції оптимальних вагових коефіцієнтів ШНМ, представлена на рис. 4.

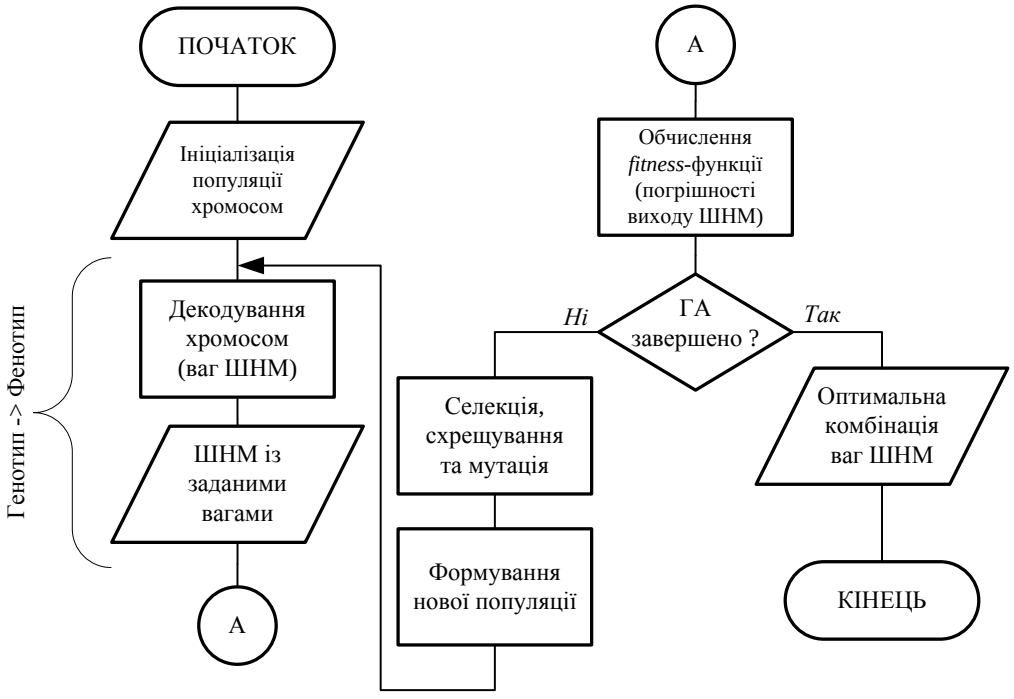

Рис. 4. Блок-схема ГА еволюційного пошуку оптимальних ваг ШНМ

Відповідно до першого етапу типового ГА апріорі задаються і залишаються незмінними архітектура ШНМ, що визначає кількість шарів, число нейронів в кожному шарі і топологію міжнейронних зв'язків, а також правило навчання ШНМ. Пристосованість кожної особини (генотипу) оцінюється значенням середньоквадратичної погрішності, розрахованої по відповідній цій особині ШНМ (фенотипу).

У представленому процесі еволюційного навчання реалізується режим так званого пакетного навчання (*batch training mode*), при якому значення ваг змінюються тільки після пред'явлення ШНМ всіх навчальних образів. Такий прийом відрізняється від вживаного в більшості послідовних алгоритмів навчання – наприклад, в методі зворотного розповсюдження помилки, ваги уточнюються після пред'явлення ШНМ кожної навчальної вибірки.

На першому етапі ГА для навчання, необхідно визначити схему представлення ваг ШНМ, тобто вибрати між бінарним представленням і кодуванням ваг дійсними числами.

Якщо для запису кожної з ваг використовується дуже мало біт, то навчання може продовжуватися дуже довго і не принести ніякого ефекту, оскільки точність апроксимації окремих комбінацій дійсних значень ваг дискретними значеннями часто виявляється недостатнім.

З іншого боку, якщо використовується дуже багато біт, то двійкові послідовності, що представляють ШНМ великої розмірності, виявляються дуже довгими, що сильно подовжує процес еволюції і робить ГА навчання ШНМ нераціональним з практичної точки зору.

5. Програмна модель навчання нейронної мережі в LabVIEW

ШНМ, що реалізує логічну функцію XOR представлена на рис. 3. Для цієї ШНМ необхідно підібрати оптимальні ваги, що мінімізують значення погрішності

$$Q = \frac{1}{4} \cdot \sum_{i=1}^{4} (d_i - y_i)^2 , \qquad (1)$$

де 4 – це кількість унікальних значень виходу ШНМ, y_i – поточний вихід ШНМ, а d_i – еталонний вихід ШНМ. ШНМ має набувати вихідних значень відповідно до табл. 1.

У якості параметрів даного завдання виступають перераховані вище вагові коефіцієнти ШНМ, тобто завдання має 6 параметрів. У даному прикладі набір з цих 6 параметрів визначає точку простору пошуку і, отже, існує можливий розв'язок завдання – навчання ШНМ (див. рис. 5).

Припустимо, що ваги можуть набувати значень з інтервалу [-1, 1]. Тоді кожна хромосома буде комбінацією з 6 двійкових (бінарних) послідовностей, що кодують конкретні ваги ШНМ. Це, очевидно, і є генотипи. Відповідні їм фенотипи представлені значеннями окремих ваг ШНМ, тобто набором (множиною) відповідних дійсних чисел також з інтервалу [-1, 1].

Хромосоми і генотипи позначають одне і те ж – фенотипи особин популяції, закодовані у формі впорядкованих послідовностей генів із значеннями (аллелями), що дорівнюють 0 або 1.

За допомогою ГА програмними засобами LabVIEW необхідно знайти оптимальний набір вагових коефіцієнтів ШНМ, а саме значення змінних: w_{11}, w_{12}, w_{21}, w_{22}, v_1, v_2, що будуть максимально наближатись до еталонних значень ваг (див. рис. 2). Тобто необхідно знайти такі значення ваг ШНМ, для яких *fitness*-функція погрішності Q (1) досягає мінімального значення, яке дорівнюватиме 0 (максимальне 1).

До платформи cRIO в мікросхему ПЛІС завантажується лише скомпільована ШНМ із невідомими ваговими коефіцієнтами. Процедуру навчання ШНМ виконує спеціальний програмний засіб в середовищі LabVIEW на платформі розробника. Це персональний ПК, на якому компілювалась структура ШНМ для ПЛІС.

У відповідності до ГА, навчання ШНМ відбувається наступним чином.

Спочатку генеруються випадкові набори вагових коефіцієнтів (пакет). Далі ці пакети у вигляді одновимірних масивів подаються на вхід параметрів (ваг) ШНМ. Також до ШНМ подаються бінарні значення вхідних та еталонних вихідних значень логічної функції XOR.

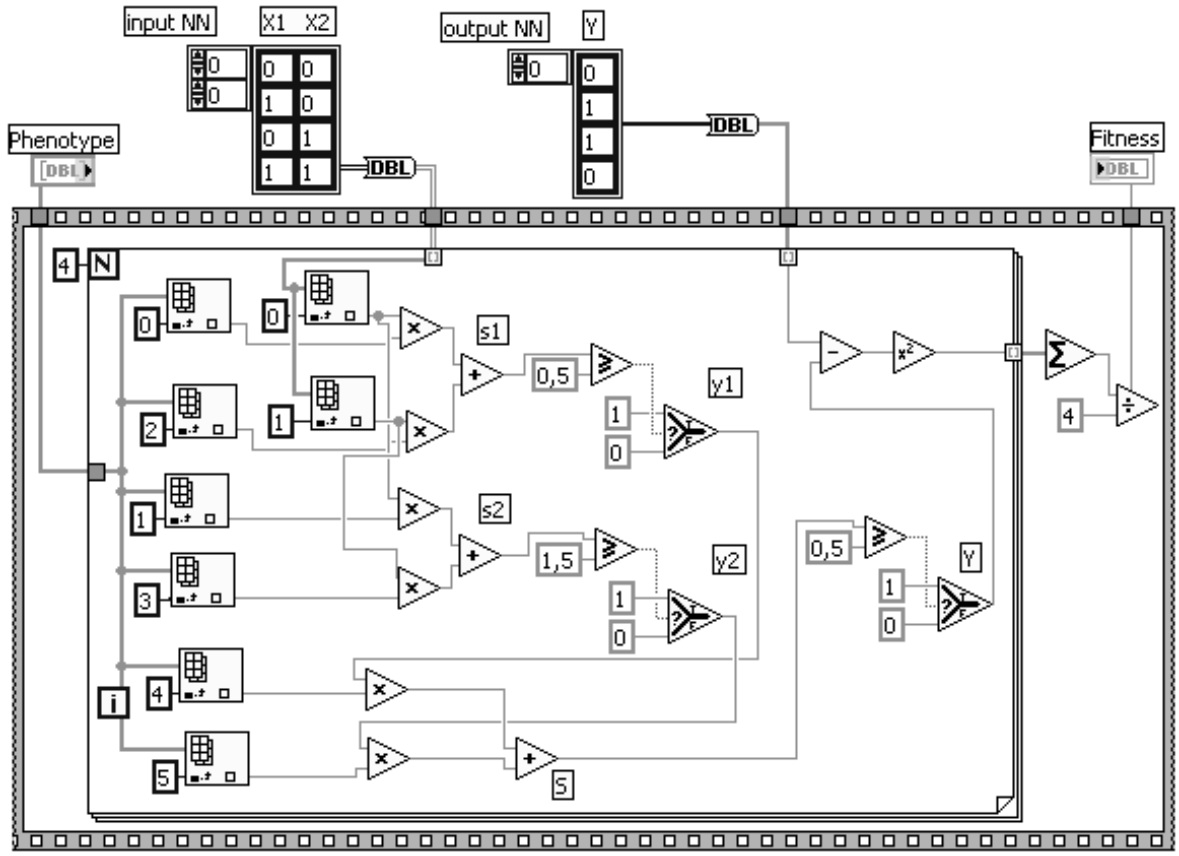

Рис. 5. Блок-діаграма *fitness*-функції в LabVIEW

Далі обраховуються пакети *fitness*-функцій погрішності Q й обирається найкраща. У відповідності до цього, надалі виконується еволюційна математика селекції, схрещування та мутації найкращих хромосом популяції, у результаті чого отримуються нові пакети хромосом вагових коефіцієнтів і процедура навчання поновлюється. Ці дії виконуються до тих пір, доки *fitness*-функція погрішності Q не досягне оптимального значення (в нашому випадку значення 0), або не вичерпається задана початкова кількість популяцій.

6. Результати навчання нейронної мережі генетичним алгоритмом

В дослідженні застосовується ГА з турнірною селекцією в підгрупах по дві особини з однією точкою схрещування та непомітною мутацією (*creep mutation*). За замовчуванням прийняті значення вірогідності схрещування 0,85 і мутації 0,05, а також розмірність популяції, яка дорівнює 50. Максимальна кількість поколінь – 500. Довжина хромосом для вирішуваного завдання дорівнює 66 бітам – по 11 генів на кожний ваговий коефіцієнт (11 біт).

Задані параметри моделювання процедури пошуку та результати пошуку оптимальних ваг ШНМ можна спостерігати на лицьовій панелі в LabVIEW, що зображена на рис. 6.

В процесі навчання із використанням ГА, було знайдено оптимальні вагові коефіцієнти ШНМ, які відповідають еталонним значенням (див. рис. 2), тобто всі ваги дорівнюють 1, окрім останнього (-1). Кількість використаних поколінь мутацій склала 4. Ці результати можна спостерігати на лицьовій панелі в полі *Solution*.

При кожній наступній спробі перенавчання ШНМ ГА, за незмінних початкових умов, спостерігається стійке навчання ШНМ протягом 3-10 поколінь. Розкид значень вагових коефіцієнтів відносно еталонних не перевищує 0.1 %, що ніяк не впливає на точність роботи даної ШНМ.

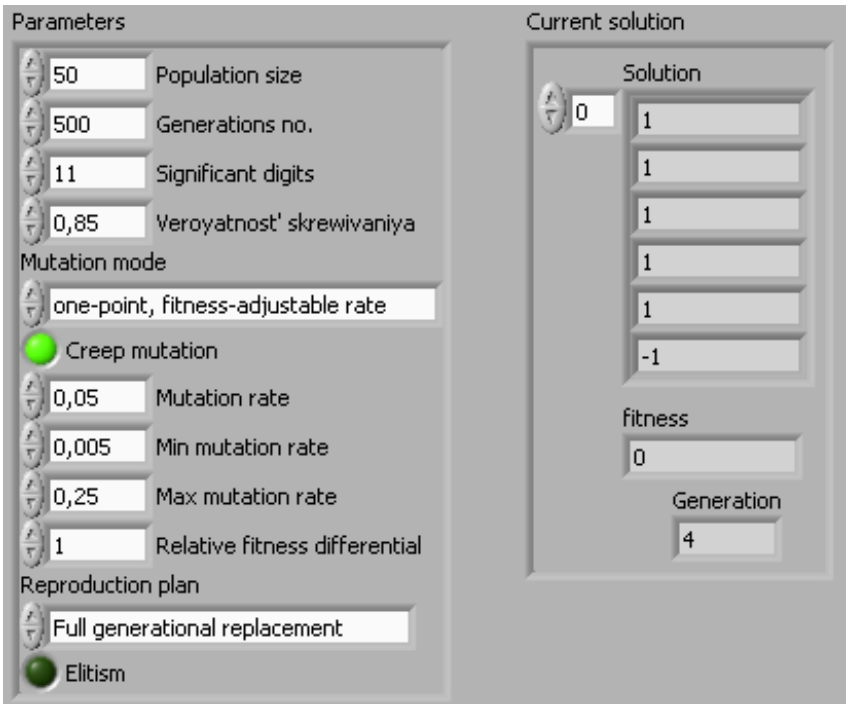

Рис. 6. Результати навчання ШНМ, що виконує логічну функцію XOR в LabVIEW

Висновки

У роботі представлений приклад проектування ШНМ із архітектурою персептрон та синтезу оптимальних ваг цієї ШНМ засобами програмування в середовищі LabVIEW. Була побудована модель ШНМ, що реалізує логічну функцію XOR.

Для програмної реалізації персептрона було обрано структуру та архітектуру ШНМ із трьома нейронами та заданими пороговими значеннями активаційної функції усіх нейронів.

Процедура навчання ШНМ відбувалась шляхом використання й реалізації класичного ГА з турнірною селекцією.

Наведемо два найважливіших аргументи на користь застосування ГА для оптимізації ваг ШНМ. Перш за все, ГА забезпечують глобальний перегляд простору ваг і дозволяють уникати локальні мінімуми. Крім того, вони можуть використовуватися в задачах, для яких інформацію про градієнтах отримати дуже складно або вона виявляється занадто дорогою.

В даному прикладі, для навчання ШНМ досить було виконати 3–15 генерацій за допомогою ГА. Алгоритмам зворотного поширення потрібно була на багато більша кількість ітерацій – близько 500 ітерацій, де одна ітерація – це повний перерахунок всіх навчальних даних (вагові коефіцієнти, похибка, значення виходів ШНМ). Зауважимо, що при цьому дві генерації в ГА еквівалентні одній ітерації алгоритму зворотного розповсюдження, оскільки зворотне розповсюдження на даному навчальному прикладі складається з двох частин – прямий прохід (обчислення виходу ШНМ і помилки) і зворотний прохід (налаштування ваг). ГА виконує лише першу частину. Таким чином, дві генерації займають менше часу, ніж обчислення однієї ітерації алгоритму зворотного розповсюдження.

1. *Хайкин С.* Нейронные сети: полный курс.: 2-е издание [пер. с англ.] / М.: Издательский дом «Вильямс», 2006. – 1104 с.
2. *Курейчик В.М.* Генетические алгоритмы и их применение // Издательство ТРТУ, издание второе, 2002. – 242 с.
3. *Каширина И.Л.* Нейросетевые технологии. Учебно-методическое пособие для вузов. / Воронеж: Изд-во ВГУ, 2008. – 72 с.
4. *Staffer J.D., Whitley L, Eshelman J.* Combinations of Genetic Algorithms and

Neural Networks: A Survey of the State of the Art, Proceedings of International Workshop on Combinations of Genetic Algorithms and Neural Networks. COGANN-92. – 1992.
5. *Рутковская Д., Пилиньский М., Рутковский Л.* Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польск. И.Д. Рудинского. – М.: Горячая линия – Телеком, 2006. – 452 с.
6. *Галушкин А.* Нейронные сети. Основы теории. – М.: Высш. шк., 2010. – 496 с.
7. *Жеребко В.А, Кравець П.І., Василевська Х.С., Степанчук О.О.* Підхід до побудови штучних нейронних мереж та їх реалізація на апаратній платформі CompactRIO // Вісник вінницького політехнічного інституту. – 2009. – № 3 (84). – С. 66–70.
8. *Жеребко В.А., Онацький А.В., Соломонюк Ю.Р.* Дослідження витрат ресурсів FPGA при побудові нейромережевої логічної функції XOR засобами LabVIEW // Обчислювальний інтелект ОІ-2011: І Міжнар. наук. конф., 10–13 травня 2011 р.: зб. пр. – ЧДТУ, 2011. – С. 165.

Одержано 10.12.2012

Про автора:

Жеребко Валерій Анатолійович,
молодший науковий співробітник НДІ ІП,
здобувач кафедри АУТС НТУУ «КПІ».

Місце роботи автора:

Національний технічний університет України «КПІ»,
03056, Київ,
Проспект Перемоги 37,
корпус 18, к. 522,
Тел. +380(44) 406 8346.
E-mail: zherebko@kpi.ua