УДК 519 – 7

Калініна І.О.

ДОСЛІДЖЕННЯ АЛГОРИТМІВ НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ У ЗАДАЧАХ ПРОГНОЗУВАННЯ

У статті досліджуються алгоритми навчання штучних нейронних мереж у задачі прогнозування. Для дослідження були обрані такі широко використовувані алгоритми навчання штучних нейронних мереж, як алгоритм генетичної селекції, алгоритм спряжених градієнтів та алгоритм зворотного поширення помилки.

Ключові слова: *прогнозування, алгоритм генетичної селекції, алгоритм спряжених градієнтів, алгоритм зворотного поширення помилки.*

В статье исследуются алгоритмы обучения искусственных нейронных сетей в задаче прогнозирования. Для исследования были выбраны такие широко-используемые алгоритмы обучения искусственных нейронных сетей, как алгоритм генетической селекции, алгоритм сопряженных градиентов и алгоритм обратного распостранения ошибок.

Ключевые слова: *прогнозирование, алгоритм генетической селекции, алгоритм сопряженных градиентов, и алгоритм обратного распостранения ошибок.*

In this article the algorithms of teaching of artificial neuron networks are probed in the task of prognostication. For research such widely utillized algorithms of teaching of artificial neuron networks, as algorithm of genetic selection, algorithm of conjugating gradients and algorithm of reverse distribution of error, were select.

Key words: *the prognostication, algorithm of genetic selection, algorithm of conjugating gradients, backpropagation algorithm.*

ВСТУП

В останні роки у світі бурхливо розвивається нова прикладна область штучного інтелекту, яка спеціалізується на використанні для вирішення інтелектуальних задач штучних нейронних мережах (ШНМ). Актуальність досліджень у цьому напрямку підтверджується величезною кількістю різноманітних використань ШНМ. Основними класами задач, у яких на сьогодні успішно використовуються рішення на базі ШНМ, є:

- задачі апроксимації;
- прогнозування;
- класифікація та розпізнавання образів;
- кластеризація;
- ідентифікація та оцінювання;
- асоціативне керування.

Із наведеного переліку прогнозування є одним із найбільш необхідних, але при цьому й одним із найскладніших завдань інтелектуального аналізу даних [3, 5, 10]. Проблеми прогнозування пов’язані з недостатньою якістю й кількістю вхідних даних, змінами середовища, у якому протікає процес, впливом суб’єктивних факторів. Прогноз завжди

здійснюється з деякою похибкою, що залежить від використовуваної моделі прогнозу й повноти вхідних даних.

Задачі прогнозування в останній час набули особливої актуальності у сфері економіки з огляду на нестабільність її – і не тільки в нашій державі, а й у всьому світі. Різкі коливання курсів валют та дорогоцінних металів вимагають більш ефективних засобів прогнозування, ніж ті, що використовуються традиційно. Є успішні рішення з побудови прогнозу із використанням імовірнісних методів і суб'єктивних знань експертів. Зокрема, дослідженням у даній області присвячені роботи [1, 2, 7].

Недоліками «класичних»методів прогнозування є:

- відсутність у моделі уявлень щодо структури й системи зв'язків реального об'єкта, що вносить суб'єктивізм у вибір як самої моделі, так і її структури;
- труднощі побудови моделей за умови, що дані зберігаються в різних часових рядах та (або) мають тимчасові зрушення стосовно один одного;
- недостатня точність прогнозу;
- значна чутливість отриманих результатів до недостатньої інформації та (або) її зашумленість;
- потреба у високій кваліфікації математиків-програмістів;
- залежність результату прогнозу від кваліфікації аналітика в конкретній предметній області.

У даній ситуації ідеальним рішенням є застосування для прогнозування часових рядів математичних моделей, заснованих на використанні апарата ШНМ, що включає в себе розвинену методологію структурного моделювання й методів навчання, основаних на добре розвинутій теорії нелінійного програмування [6, 8].

Однак використання ШНМ для рішення завдань прогнозування часових рядів і дотепер недостатньо розвинене – і в теоретичному, і в практичному аспектах. Так, наприклад, досі немає однозначної думки щодо використання тих чи інших алгоритмів навчання ШНМ, що, безумовно, є дуже важливим питанням з огляду на те, що вибір правильного алгоритму навчання фактично і визначає точність прогнозування.

В якості прикладу задачі прогнозування розглянемо задачу прогнозування курсу дорогоцінних металів. Задача прогнозування курсу дорогоцінних металів відноситься до класу задач прогнозування часових рядів. Справді, існує часовий ряд $\{t_1, t_2, \ldots, t_n, \ldots\}$ статистичних даних безпосередньо курсу дорогоцінного металу та факторів, що впливають на цей курс (наприклад, курс доллара). З нього вибирається кінцеве число членів $t_1, \ldots, t_n$,. Потрібно скласти таку функцію $f(x_1, \ldots x_n)$, $k \le n$, щоб для неї виконувалося наступне:

$$\left| f(t_1 \ldots t_{i+k-1}) - t_{i+k} \right|,$$

тобто функція f (її значення) є «гарним» (у деякому змісті) прогнозом значення t_{i+k} за значенням часового вікна $\{ t_1, \ldots, t_{i+k-1} \}$.

Оскільки критерій якості прогнозу може бути різним, то в даному випадку він не обговорюється. Кінцевою метою задачі є її використання для прогнозу невідомих значень t_{n+1}, $t_{n+2}, \ldots$, оскільки передбачається, що якщо для великої кількості часових вікон знайдений «гарний» прогноз, то він залишається «гарним» і для невеликої кількості часових вікон, що виходять за границі відомих значень.

На рис. 1 проілюстровано обчислення прогнозної функції. Кожний вузол мережі приймає на вхід множину вихідних значень попереднього шару мережі та передає результат обчислень на наступний шар таким чином, що:

a_k^t – коефіцієнт зв'язку між i-м вузлом (L-1)-го шару чи з k-м вузлом i-го шару, а $y^{L-1} = f\left(a_i^{L-1}\right)$ – результат обчислення у вузлі номер i шару номер L-1.

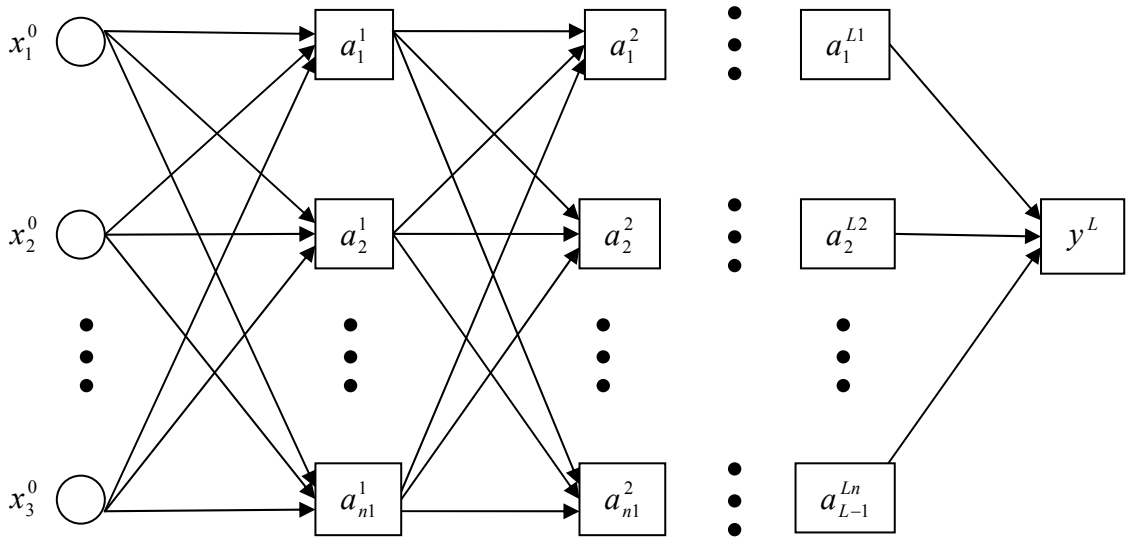

Рис. 1. Обчислення функції прогнозу

Таким чином, нейронна мережа – однорідна композиція шарів елементарних обчислювачів. З боку тільки функції прогнозу абсолютно все рівно, з якого інтервалу брати значення: [0; 1] чи [-1; 1]. Зазвичай прийнято брати [0; 1], але експерименти показують, що в деяких випадках [-1; 1] є доцільнішим.

Оскільки величини елементів часового ряду можуть змінюватися безперервно (на відрізку), то й вся мережа повинна постійно знаходитися в безперервному режимі, щоб реалізувати безперервну за всіма аргументами функцію $f_0(.)$. Тому вважається небажаним входження вузла (елементарного обчислювача) до режиму «насичення», тобто потрапляння a_i^L до такої точки, де $f'\left(a_i^L\right)$ – перша похідна – досить мала.

Оскільки розглядаються цілком визначені функції, то це означає, що величина a_i^L не повинна сильно відхилятися від нуля.

Розглянемо обмеження $a^- \le a_i^L \le a^+$, $\left|a^-\right| = \left|a^+\right|$. Якщо $\left|a^+\right|$ мала, то $f(x)$ досить точно апроксимується лінійною функцією.

Навчання мережі – це процес пошуку таких коефіцієнтів зв'язку, при якому нейронна мережа реалізує «гарну» функцію прогнозу. Оскільки мережа реалізує безперервну функцію, то вона може прогнозувати так само як декілька нормований ряд, так і перетворений, наприклад, коли всі значення зменшені в k разів. Досить зручно елементарний обчислювач є функцією вигляду (4), оскільки в цьому випадку, якщо усі входи дорівнюють нулю, то й вихід також буде дорівнювати нулю. Разом з цією функцією потрібно дотримуватися ще однієї умови (для більшої надійності обчислень) – середня величина часового ряду повинна дорівнювати нулю.

ЗАДАЧІ ДОСЛІДЖЕННЯ

Необхідно дослідити алгоритми навчання штучних нейронних мереж при вирішенні задачі прогнозування та визначити найбільш ефективний алгоритм навчання.

ДОСЛІДЖЕННЯ АЛГОРИТМІВ НАВЧАННЯ

Розглядаються три групи алгоритмів навчання нейронних мереж: алгоритми генетичної селекції, алгоритми градієнтного спуску, алгоритм зворотнього розповсюдження помилки.

Алгоритм генетичної селекції. Генетичні алгоритми навчання штучних нейронних мереж базуються на теоретичних досягненнях синтетичної теорії еволюції, яка враховує мікробіологічні механізми спадкування ознак у природних і штучних популяціях організмів, а також на накопиченому людському досвіді в селекції тварин і рослин. При цьому механізмами зміни визначених ознак виступають схрещування (гібридизація) та мутації.

Методологічна основа генетичних алгоритмів ґрунтується на гіпотезі селекції, яка в самому загальному вигляді може буди сформульована таким чином: чим вища пристосованість особини, тим вища імовірність того, що в потомстві, отриманому з її участю, ознаки, які визначають пристосованість, будуть виражені ще більше.

Оскільки генетичні алгоритми мають справу з популяціями постійної чисельності, тут особливої актуальності нарівні з селекцією у батьків набуває відбір на знищення. Найчастіше особини, які володіють низькою пристосованістю, не тільки не беруть участі у генерації нового покоління, а й вилучаються з популяції на черговому дискретному кроці еволюції.

Вхідні дані подаються у вигляді вектора. Координати векторів несуть змістовну «генетичну» інформацію. Подібно до того, як у природі схрещування організмів здійснюється на генетичному рівні, у процедурі оптимізації координати нових спробних точок отримуються як результат маніпулювання координатами старих. Відмінною рисою генетичного алгоритму від інших методів навчання штучних нейронних мереж є те, що він дозволяє відшукати глобальний мінімум похибки при налаштуванні ваг, хоча з малою точністю.

У загальному випадку генетичний алгоритм відрізняється від інших чисельних методів оптимізації тим, що запозичає з біології:

- понятійний апарат;
- ідею колективного пошуку екстремуму за допомогою популяції особин;
- способи подання генетичної інформації;
- способи передачі генетичної інформації в черзі поколінь (генетичні оператори);
- ідею про переважність розмноження найбільш пристосованих особин.

Вектор вхідних і вихідних ваг нейронної мережі $W = (W_{inp}, W_{mid}^{j}, W_{out})$, де W_{inp} – вектор вхідних ваг нейронної мережі; W_{mid}^{j} – вектор ваг між j-м і (j + 1) – м прихованими шарами нейронної мережі (для багатошарової нейронної мережі); $j = \overline{1..k-1}$, де k – число прихованих шарів; W_{out} – вектор вихідних ваг нейронної мережі.

Розмірність W дорівнює $(N + 1)\ J_1 + J_1J_2 + .. + J_{k-1}J_k + J_kM$, де N – число входів НМ; J_j – число нейронів *j-го* прихованого шару; M – число виходів НМ.

Навчання нейронної мережі за допомогою генетичного алгоритму – ітеративний процес відшукання такого набору ваг нейронної мережі за допомогою випадкової зміни компонентів, яка б задовольняла заданий критерій навчання.

За критерій навчання візьмемо суму квадратів відхилень, отриманих за моделлю виходів і бажаних виходах.

$$e(W) = \frac{1}{2}\left\|d - y(W)\right\|^2 = \frac{1}{2}\sum_{k=1}^{M}(d_k - y_k(W))^2 .$$

Процес навчання припиняється при досягненні критерієм заданого порогу для будь-якого з шуканого набору ваг. На практиці точність налаштування ваг за допомогою генетичного алгоритму невисока й задана точність (порядку 0,001 чи вище) не досягається. Тому процес навчання припиняється після закінчення заданої кількості кроків еволюції. Кращий набір ваг на цей момент є шуканим.

У роботі величина навчальної послідовності визначається при побудові моделі вхідних даних для кожної прогнозованої точки. Генетичний алгоритм призначений для навчання нейронної мережі шляхом налаштування її вхідних і вихідних ваг. У результати навчання виходи нейронної мережі повинні збігатися з бажаними виходами (реальними даними) із заданим ступенем вірогідності (критерій навчання).

Процедуру навчання ШНМ за допомогою ГА можна зобразити на блок-схемі наступним чином:

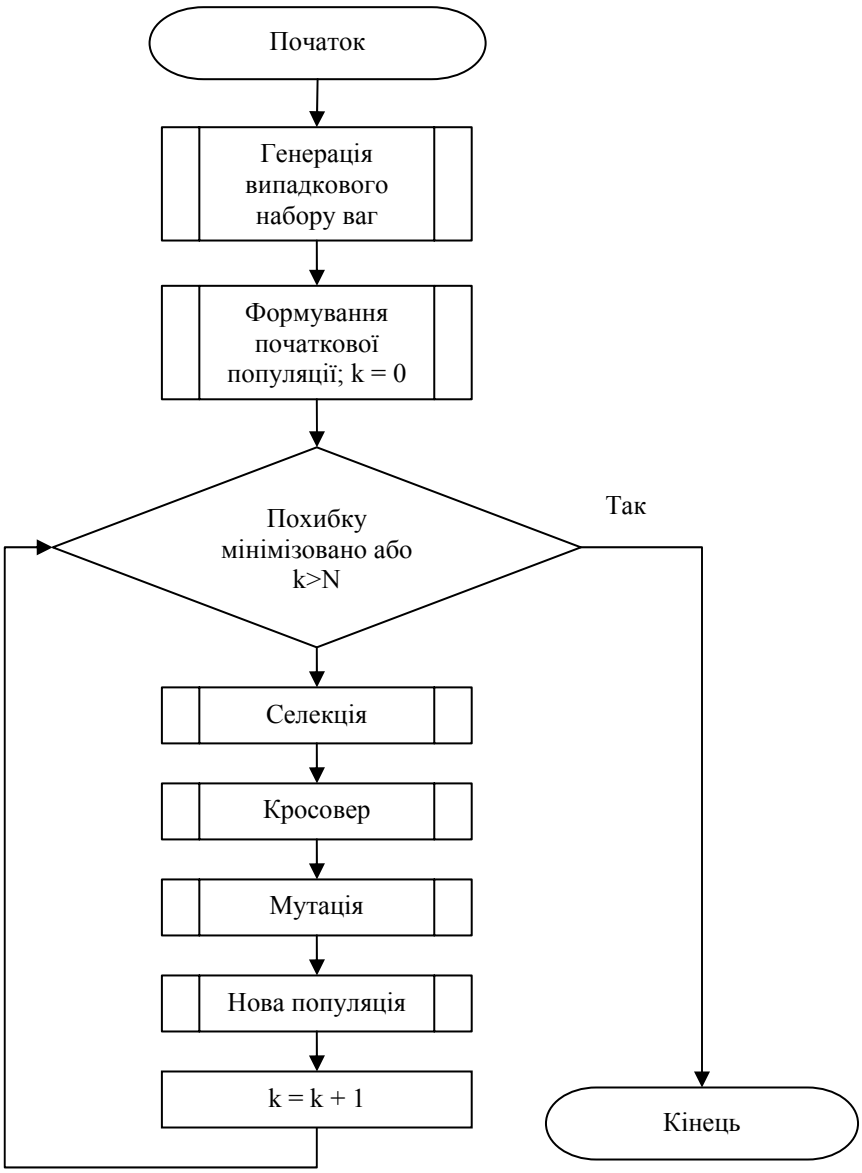

Рис. 2. Блок-схема навчального алгоритму генетичної селекції

Алгоритми з використанням методів градієнтного спуску. Задачу навчання мережі у випадку застосування градієнтних алгоритмів розглядають як задачу мінімізації апріорно визначеної цільової функції *ξ(w)*. Градієнтні методи зв'язані з розкладанням цільової функції *ξ(w)* в ряд Тейлора в найближчому околі точки наявного рішення *w*. У випадку цільової функції від багатьох змінних ($w = [w_1, w_2, ..., w_n]^T$) таке представлення пов'язується з околом раніше визначеної точки (при старті алгоритму – це вихідна точка w_0) в напрямку *x*. Подібне розкладання описується універсальною формулою виду:

$$\xi(w+x) = \xi(w) + [p(w)]^T x + \frac{1}{2} x^T H(w) x + \dots,$$

де $p(w) = \nabla \xi = \left[\frac{\delta \xi}{\delta w_1}, \frac{\delta \xi}{\delta w_2}, \dots, \frac{\delta \xi}{\delta w_n} \right]^T$ – це вектор градієнта, а симетрична квадратна матриця

$$H(w) = \begin{bmatrix} \frac{\delta^2 \xi}{\delta w_1 \delta w_1} & \dots & \frac{\delta^2 \xi}{\delta w_1 \delta w_n} \\ \dots & \dots & \dots \\ \frac{\delta^2 \xi}{\delta w_{1n} \delta w_1} & \dots & \frac{\delta^2 \xi}{\delta w_n \delta w_{1n}} \end{bmatrix}$$

є матрицею похідних другого порядку і називається гессіаном.

X відіграє роль направляючого вектора, що залежить від фактичних значень вектора *w*. На практиці частіше розраховуються три перших члени ряду, а наступні просто ігноруються.

Для спрощення опису значення змінних, одержані в *k*-му циклі, будемо записувати з нижнім індексом *k*. Точкою рішення $w = w_k$ будемо вважати точку, в якій досягається мінімум цільової функції $\xi(w)$ та $p(w_k)=0$, а гессіан $H(w_k)$ є невід'ємно визначеним. При виконані цих умов функція в будь-якій точці, що лежить в околі w_k, буде мати більше значення, ніж у точці w_k, тому точка w_k є рішенням, що відповідає критерію мінімізації цільової функції.

У процесі пошуку мінімального значення цільової функції напрямок пошуку *p* та крок *h* підбираються таким чином, щоб для кожної наступної точки $w_{k+1} = w_k + \eta_k x_k$ виконувалася умова $\xi(w_{k+1}) < \xi(w_k)$. Пошук мінімуму виконується до тих пір, поки норма градієнта не опуститься нижче апріорі заданого значення допустимої похибки, або поки не буде перевищено максимальної кількості ітерацій. Універсальний оптимізаційний алгоритм навчання нейронної мережі можна представити в наступному вигляді (будемо вважати, що початкове значення вектора, що оптимізується, відоме і складає $w_k = w_0$):

Крок 1: Перевірка зходжуваності та оптимальності поточного значення w_k. Якщо точка w_k відповідає градієнтним умовам зупинки процесу – завершення обчислень. У протилежному випадку перейти до кроку 2.

Крок 2: Визначення вектора напрямку оптимізації x_k для точки w_k.

Крок 3: Вибір величини кроку η_k у напрямку x_k, при якому виконується умова $\xi(w_k + \eta_k x_k) < \xi(w_k)$.

Крок 4: Визначення нового рішення $w_{k+1} = w_k + \eta_k x_k$, а також відповідних йому значень $\xi(w)$ та $p(w_k)$, а якщо треба, то і $H(w_k)$, та повернення до кроку 1.

Алгоритм спряжених градієнтів. Цей алгоритм відрізняється від загального алгоритму градієнтного спуску тим, що при виборі напрямку мінімізації не використовується інформація про гессіан. Напрям пошуку x_k вибирається таким чином, щоб він був ортогональним та спряженим до всіх попередніх напрямків $x_0, x_1, \ldots, x_{k-1}$. Множина векторів x_i, $i = 0, 1, \ldots, k$ буде взаємно спряженою стосовно матриці ***A***, якщо $x_i^T A p_j = 0, i \neq j$ (1)

Вектор, що задовольняє задані умови, має вигляд:

$$x_{k+1} = x_k + \alpha_k p_k, \qquad (2)$$

де $p_k = p(w_k)$ є фактичним значенням вектору градієнта.

Із формули (2) впливає, що новий напрямок мінімізації залежить тільки від значення градієнта в точці рішення w_k та від попереднього напрямку пошуку p_k, помноженого на коефіцієнт спряження α_k.

Алгоритм зворотного поширення помилки. В загальному вигляді алгоритм зворотного поширення помилки являє собою наступну послідовність кроків:

Крок 1: Ініціювати ваги малими випадковими величинами.

Крок 2: Якщо умова зупинки не виконується, виконати кроки 3-10.

Крок 3: Для кожної навчальної пари виконати кроки 4-9.

Прямий прохід:

Крок 4: Кожен вхідний нейрон $x_i = 1 \ldots n$ приймає вхідний сигнал і поширює його до всіх нейронів прихованого шару.

Крок 5: Кожен нейрон прихованого шару $v_j = 1 \ldots q$ підсумовує свої зважені вхідні сигнали: $h_j = \sum_i^n w_{ij} x_i$, застосовує до одержаної суми функцію активації, формуючи вихідний сигнал: $v_j = f(h_j)$, котрий надсилається до всіх нейронів вихідного шару.

Крок 6: Кожен вихідний нейрон y_k, $k = 1 \ldots m$ підсумовує зважені сигнали: $h_k = \sum_j^q w_{jk} v_j$, формуючи після застосування функції активації вихідний сигнал мережі: $y_k = f'(h_k)$.

Зворотне поширення помилки:

Крок 7: Кожен вихідний нейрон співставляє своє значення виходу з потрібною цільовою функцією і вираховує $\delta_k = (t_k - y_k) f'(h_k)$, після чого визначається корегуючий член ваг: $\Delta w_{jk} = \eta \delta_k v_j$, а параметри δ_k надсилаються в нейрони прихованого шару.

Крок 8: Кожен нейрон прихованого шару v_j підсумовує свої δ-входи від нейронів вихідного шару: $h_k = \sum_k^m \delta_k w_{jk}$, результат помножують на похідну від функції активації для визначення δ_j: $\delta_j = f'(h_j) \sum_k^m \delta_k w_{jk}$

та вираховується корегуючий член: $\Delta w_{ij} = \eta \delta_k w_{jk}$.

Корегування ваг:

Крок 9: Ваги між прихованим та вихідним шарами модифікуються так:

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}.$$

Аналогічно корегуються ваги між вхідним та прихованим шарами:

$$w_{ij}(new) = w_{ij}(old) + \Delta w_{ij}.$$

Крок 10: Перевіряється умова зупинки: мінімізація похибки між потрібним та реальним виходом мережі.

За допомогою блок-схеми поданий алгоритм можна зобразити наступним чином (рис. 3).

Порівняльний аналіз алгоритмів навчання штучної нейронної мережі

Порівняльний аналіз обраних для дослідження алгоритмів проводиться за такими критеріями, як час виконання алгоритму, час виконання програмної реалізації алгоритмів, швидкість сходження алгоритмів та точність результату програми при заданій кількості ітерацій.

Часова складність. Час виконання алгоритму являє собою кількість кроків, які виконуються в даному алгоритмі від початку до кінця. Для підрахування кількості кроків досліджуваних у даній роботі алгоритмів, зручно скористатися наведеними вище блок-схемами.

Результати підрахунків наведено у таблиці 1:

Таблиця 1

Час виконання алгоритмів навчання ШНМ

Назва алгоритму	**Час виконання алгоритму (кроки)**
Алгоритм генетичної селекції	8
Алгоритм спряжених градієнтів	11
Алгоритм зворотного поширення помилки	16

Кількість кроків характеризує собою ніщо інше, як складність реалізації алгоритму, що є зараз досить важливим фактором, адже часто робочий час програміста коштує дорожче, ніж робочий час комп'ютера, на відміну від минулих років.

На швидкість виконання програми в цілому впливають такі фактори:

- введення інформації в систему;
- якість скомпільованого коду програми;
- часова складність програми.

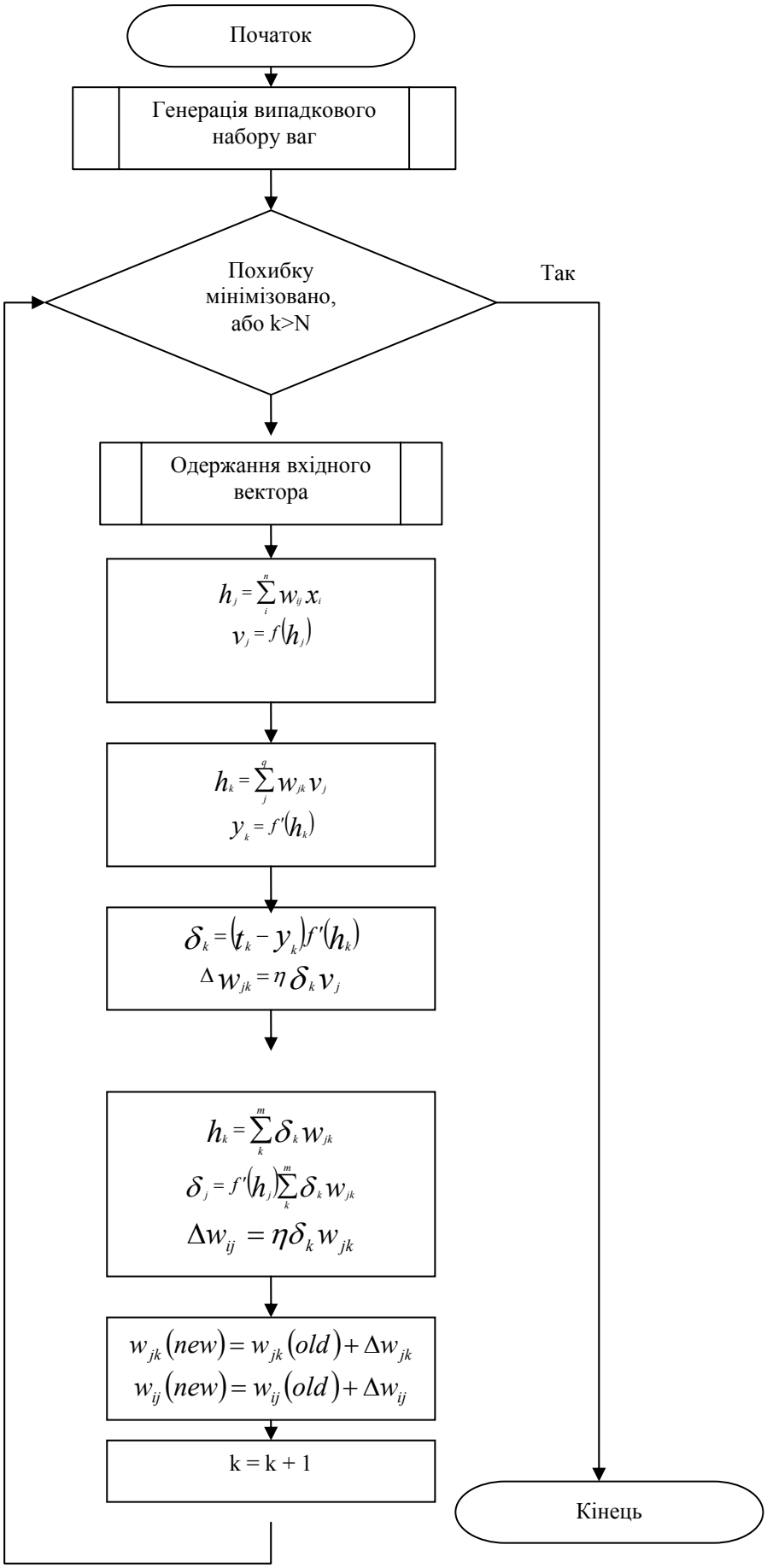

Рис. 3. Блок-схема алгоритму зворотного розповсюдження помилки

Оскільки програмні реалізації всіх трьох алгоритмів навчання ШНМ оперують з одним і тим самим набором вхідних даних, використовуються для навчання однієї й тієї ж структури ШНМ та скомпільовані одним компілятором, то першими двома факторами можна знехтувати. Таким чином, нас цікавить лише часова складність програмної реалізації кожного з досліджуваних алгоритмів навчання ШНМ.

Часову складність програми визначають за допомогою так званої О-нотації. Асимптотична нотація великого О, відома також як нотація Ландау, – розповсюджена математична нотація для формального запису асимптотичної поведінки функцій. Широко вживається в теорії складності обчислень, інформатиці та математиці.

Оскільки всі досліджувані в даній роботі алгоритми оброблюють один і той же набір вхідних даних, цю величину можна прийняти константною й такою, що не відіграє суттєвої ролі у встановленні часової складності програмної реалізації кожного з вищенаведених алгоритмів.

Таким чином, із величин, що мають значення при виконанні алгоритму навчання ШНМ: набір навчальних даних, набір ваг та кількість ітерацій навчання, залишаються лише останні дві. Отже, кількість ітерацій (k) і розмір вектора вагів (w) будуть виконувати роль міри об'єму вхідних даних.

Обчислимо часову складність першого із розглянутих алгоритмів – навчального алгоритму генетичної селекціїю

Сам цикл навчання буде виконуватися $N - 1$ разів, тому, користуючись правилом добутку, можемо обчислити загальний час виконання програми, що реалізує навчальний алгоритм генетичної селекції:

$$O_{заг} = O(N - 1) * O(w) = Nw - w.$$

Ми могли б знехтувати величиною w, якби мали на меті використовувати даний алгоритм лише для ШНМ з незначним за розміром вектором вагів. Однак для вирішення багатьох задач (і задачі прогнозування в тому числі) доводиться використовувати доволі складні структури нейронних мереж, тому відкидання числа w, як такого, що не має суттєвого впливу на час навчання, може бути недоцільним. Отже, підсумовуючи все вищесказане, одержимо, що час виконання програми, яка реалізує навчальний алгоритм генетичної селекції, буде дорівнювати $O_{ГА} = Nw - w$.

Такий же результат одержимо і у випадку із алгоритмом спряжених градієнтів, оскільки, подібно до ГА, цей алгоритм виконує свої операції відразу для всього вектора вхідних вагів і також проходить через $N - 1$ ітерацій.

Тому можемо відразу визначити, що $O_{СГ} = Nw - w$.

Із алгоритмом зворотного поширення похибки ситуація є дещо іншою. Він також виконується $N - 1$ разів, однак операції всередині основного циклу виконуються пошарово, тобто не w разів, як у попередніх двох випадках, а w/m, де m – кількість шарів вектору вагів.

Таким чином, як і раніше використовуючи правило суми та правило добутку, одержимо: $O_{ЗПП} = Nw/m - w/m$.

Ми не можемо співвідносити кількість ітерацій головного циклу алгоритму (будь-якого з вищеозначених) і розмір вектора вагів, оскільки не знаємо, наскільки ці величини відрізняються. Завжди залишається невелика ймовірність того, що вдалий набір ваг буде віднайдено ще при випадковій генерації, і в такому випадку кількість ітерацій не буде відігравати ніякої ролі. Тому в даній роботі ніяких співвідношень та спрощень щодо величин N та w не виконується.

Результати обчислень часу виконання програм, що реалізують досліджувані алгоритми навчання ШНМ, наведено у таблиці 2:

Таблиця 2

Часова складність досліджуваних алгоритмів

Програмна реалізація алгоритму	Час виконання програми
генетичної селекції	$O_{ГА} = Nw - w$
спряжених градієнтів	$O_{ГА} = Nw - w$
зворотного поширення помилки	$O_{ЗПП} = Nw/m - w/m$

Отже, можна сказати, що кожен із наведених алгоритмів є лінійним по своїй суті, а наскільки важливими є параметри N, w та m можна сказати лише при розгляді конкретної структури ШНМ.

Вибір структури мережі проводився за допомогою програми STATISTICA Neural Networks.

Навчання ШНМ алгоритмом генетичної селекції виконувалося з використанням програмного забезпечення GeneHunter.

Навчання ШНМ алгоритмами спряжених градієнтів та зворотного поширення помилки проводилося з використанням програми STATISTICA Neural Networks.

Обрану структуру нейронної мережі, яка буде ідентичною для всіх трьох алгоритмів, показано на рис. 4.

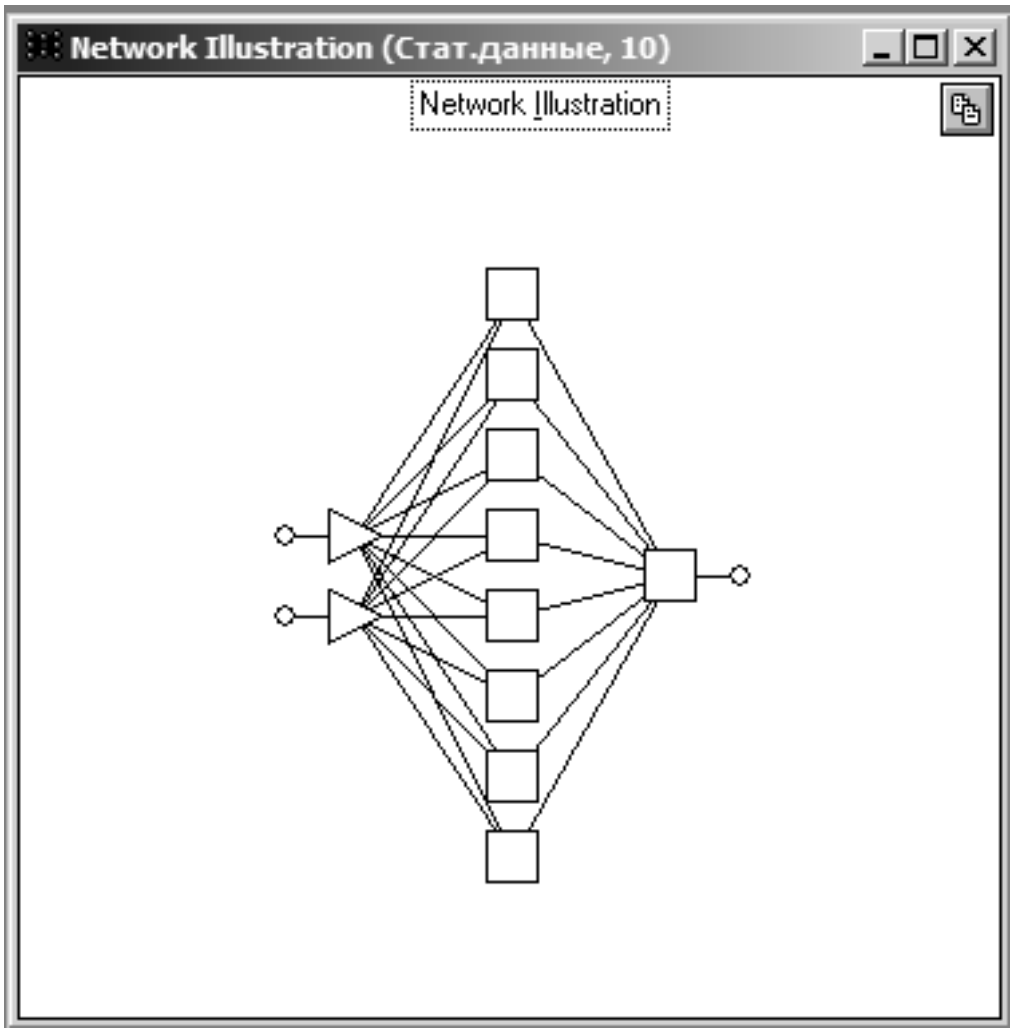

Рис. 4. Структура штучної нейронної мережі

Результати експериментальних досліджень наведено у таблиці 3:

Таблиця 3

Середня похибка прогнозування досліджуваних алгоритмів при заданій кількості ітерацій

Алгоритм	Кількість епох*	Середня похибка
Генетичний алгоритм	1 000	0.1607812
Спряжених градієнтів	1 000	0.1403673**
Зворотного поширення помилки	1 000	0.1549992

* Епохою вважається одне повне пред'явлення всіх навчальних даних.

** Навчання доводилося перезапускати кілька разів через потрапляння алгоритму до локального мінімуму.

Таблиця 4

Зведена порівняльна таблиця алгоритмів навчання ШНМ

Алгоритм/ Фактор	Алгоритм генетичної селекції	Алгоритм спряжених градієнтів	Алгоритм зворотного поширення помилки
Складність алгоритму (кроки)	8	11	16
Час виконання програми (О-нотація)	$O_{ГА} = Nw - w$	$O_{ГА} = Nw - w$	$O_{ЗПП} = Nw/m - w/m$
Похибка (10 000 епох)	0.1287812	0.1203673	0.1249992
Фактична тривалість навчання (для 10 000 епох) (с)	29 с	24 с з урахуванням затримки на перезапуск алгоритму при локальному мінімумі – 52 с.	30 с

Слід зробити пояснення до даної таблиці: як бачимо, наявна деяка невідповідність у співвідношеннях теоретичного і фактичного часу виконання програмних реалізацій досліджуваних алгоритмів навчання ШНМ.

Це пояснюється тим, що з огляду на досить невеликий розмір вхідного шару ШНМ (що пояснюється відсутністю достатніх статистичних даних), структура мережі є доволі примітивною, а тому і величина вектора вагів є незначною.

Однак слід зазначити, що при ускладненні структури мережі величина вагового вектора може стати фактором, що матиме суттєвий вплив на час навчання ШНМ.

Взагалі ж, виходячи з наведених результатів, можна зробити висновок, що оптимальним вибором для навчання ШНМ в задачі прогнозування курсу дорогоцінних металів є алгоритм зворотного поширення помилки з огляду на оптимальне співвідношення часу навчання і точності прогнозування.

Такий параметр, як складність алгоритму, відіграє менш важливу роль, оскільки у такій задачі, як прогнозування курсу дорогоцінних металів, валют та ін., головним фактором є точність прогнозу, так як ця точність прямо пропорційна до одержуваних прибутками компанії-замовника.

ВИСНОВОК

Було проведено дослідження трьох відібранних алгоритмів навчання штучних нейронних мереж – алгоритму генетичної селекції, алгоритму спряжених градієнтів та алгоритму зворотного поширення помилики.

Особливістю дослідження було те, що вказані алгоритми навчання ШНМ розглядалися в ракурсі задачі прогнозування курсу дорогоцінних металів, що зумовило пріоритетність такого критерію, як точність прогнозу навченої мережі, а також швидкість навчання з огляду на високу мінливість прогнозованої величини.

У результаті дослідження було встановлено, що найбільш оптимальним стосовно двох вищесказаних та при врахуванні усіх інших критеріїв оцінювання є алгоритм зворотного поширення помилки.

ЛІТЕРАТУРА

1. Бокс Дж., Дженкинс Г. Анализ временных рядов. Прогноз и управление. – М.: Мир, 1974.
2. Боровиков В.П., Ивченко Г.И. Прогнозирование в системе STATISTICA в среде Windows. Основы теории и интенсивная практика на компьютере.– М.: Финансы и статистика, 2000. – С. 320.
3. Боровиков В. STATISTICA для профессионалов. – СПб.: Питер. 2001. – 655 с.
4. Гамбаров Г.М., Журавель Н.М., Королев Ю.Г. Статистическое моделирование и прогнозирование: Под ред. А.Г. Гранберга. – М.: Финансы и статистика, 1990. – С. 140.
5. Методы нейроинформатики. отв. за выпуск М.Г. Доррер. –Красноярск: КГТУ, 1998. – 205 с.
6. Ежов А.А., Шумский С.А. Нейрокомпьютеринг и его применение в экономике и бизнесе – М.: Диалог-МИФИ, 1998.
7. Лобов Г.С. Методы обработки разнотипных экспериментальных данных. – Новосибирск: Наука, 1981. – 157 с.
8. Осовский С. Нейронные сети для обработки информации / Пер. с польского И.Д. Рудинского. – М.: Финансы и статистика, 2002. – 344.: ил.
9. Рутковская Д., Пилинский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польского И.Д. Рудинского. – М.: Горячая линия – Телеком, 2006. – 452.: ил.
10. Соколов Е.Н., Вайтнявичус Г.Г. Нейроинтеллект: от нейрона к нейрокомпьютеру. – М.: Наука, 1989. – С. 283.
11. Уоссермен Ф. Нейрокомпьютерная техника: теория и практика. Русский перевод: Ю.А. Зуев, В.А. Точенов. – М: Мир. – 1992.

Рецензенти: д.т.н., проф. Бідюк П.І.,
д.т.н., проф. Данілов В.Я.

Стаття надійшла до редакції 08.09.09