КІБЕРНЕТИКА та КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ

УДК 519.711 DOI:10.34229/2707-451X.24.4.3

В.А. ПЕПЕЛЯЄВ, Н.А. ОРЄХОВА, І.О. ЛУК'ЯНОВ

ОЦІНКА ЗНАЧУЩОСТІ ФАКТОРІВ КОМП'ЮТЕРНОЇ МОДЕЛІ ЗА ДОПОМОГОЮ ПРОСТОЇ НЕЙРОМЕРЕЖІ

Запропоновано алгоритм оцінки значущості факторів на основі простої нейромережі, створеної за допомогою бібліотеки Keras. Алгоритм застосовувався для виявлення малозначущих факторів у наборі початкових популяцій для дослідження багатопопуляційного генетичного алгоритму.

Ключові слова: *набір початкових популяцій, значущість факторів, нейромережа, епохи навчання.*

Вступ. Сучасний підхід у світовій практиці комп'ютерного моделювання складних стохастичних систем використовує так звану методологію Data Farming [1–5], яка базується на інтеграції засобів імітації, технологій високопродуктивних (розподілених) обчислень, методів оптимізації та інтелектуального аналізу даних. Засоби імітації надають зручну можливість створення програм, що описують структуру системи, функціональних зв'язків між її компонентами, керування системним часом, контроль послідовністю подій, що відбуваються при функціонуванні системи та ін. Методи оптимізації забезпечують можливість реалізації широкого спектру оптимізаційно-імітаційних експериментів [6–9] на багатовимірних просторах значень параметрів та характеристик досліджуваних стохастичних систем. Технології розподілених обчислень скорочують час проведення оптимізаційно-імітаційних експериментів, що надає можливість збільшити число прогонів комп'ютерних моделей на багатовимірному просторі значень параметрів за наявності часових обмежень. Методи інтелектуального аналізу даних дозволяють керувати комп'ютерним експериментом, здійснювати ефективну обробку результатів моделювання, виявляти перспективні альтернативи, забезпечують прийняття адекватних управлінських та проєктних рішень.

Існуючі засоби комп'ютерної техніки надають можливість розробляти моделі будь-якої складності. Ця обставина провокує розробників комп'ютерних моделей складних систем до зайвої деталізації. Серед досвідчених фахівців з комп'ютерного моделювання існує думка, що іноді на попередньому етапі розробки моделі кількість малозначущих факторів може досягати 80 %. Таке збільшення розмірності не тільки значно ускладнює здійснення комп'ютерних експериментів, але й може мати суттєвий вплив на розуміння взаємодії важливих факторів, які визначають основу і суть функціонування складної системи. Тому не менш важливим для подальшого модельного дослідження,

а особливо оптимізації складної системи, є визначення малозначущих факторів. Виявлення малозначущих факторів зовсім не означає, що треба заново розробляти комп'ютерну модель. Достатньо просто зафіксувати їхні значення. Тим більше, що в подальших дослідженнях може виникнути необхідність з'ясувати питання, пов'язані з деякими з цих малозначущих факторів.

Постановка задачі

В роботах [10, 11] описано та розглянуто робочі характеристики багатопопуляційного генетичного алгоритму (БГА). Згідно методології Data Farming, цей алгоритм призначено для проведення оптимізаційно-імітаційних експериментів та інтеграції можливостей технологій високопродуктивних (розподілених кластерних) обчислень та евристичного генетичного методу оптимізації. Генетичний метод ефективно працює при комп'ютерному моделюванні багатофакторних складних систем завдяки простоті у керуванні і тому входить в оптимізаційні блоки багатьох систем імітації [12].

БГА розроблявся та досліджувався на комп'ютерній платформі вітчизняного суперкомп'ютера з кластерною архітектурою СКІТ-4. В БГА виділяють три етапи. На першому етапі випадковим чином генерується початкова популяція значень факторів комп'ютерної моделі (для кожного процесора окремо). На другому – із застосуванням операцій схрещування між хромосомами-рішеннями та обмінів перспективних хромосом між різними популяціями – здійснюється основний процес оптимізації. На цьому етапі шляхом застосування визначених стратегій схрещування та обміну в деяких випадках вдається досягти 97 % від оптимуму (що іноді можна вважати досягненням квазіоптимального рішення). На третьому етапі застосуванням виключно операцій мутацій досягається оптимальне рішення.

Висока ефективність другого етапу цілком залежить від повноти «генетичного матеріалу», який отримано на першому. «Генетичний матеріал» вважається повним, якщо в початковій популяції присутні всі гени (всі значення факторів), що входять до оптимального рішення. Це досягається при генерації початкової популяції за допомогою алгоритму рівномірного сканування простору значень факторів [13, 14], коли кожна наступна хромосома-рішення генерується випадковим чином, але із простору можливих значень факторів виключаються ті, що вже присутні в попередніх хромосомах. Таким чином можна бути впевненим, що початкова популяція, розмір якої співпадає з максимальною кількістю значень серед усіх факторів, матиме повний «генетичний матеріал».

Крім того, загальним чинником, який впливає на швидкість збіжності оптимізаційних алгоритмів є розмірність. Як згадувалось вище, розмірність комп'ютерної моделі можна значно зменшити, якщо виявити і виключити із розгляду малозначущі фактори. Цілком зрозуміло, що виникає бажання зробити це якомога раніше. Раніш за все це можна зробити на основі даних, що надає набір початкових популяцій.

При дослідженні БГА розглядалася залежність його ефективності від кількості популяцій [15, 16]. Було виявлено, що максимальна обчислювальна ефективність БГА досягається при застосуванні 8–10 популяцій. Якщо максимальна кількість значень серед усіх факторів досягає 30, то загальний розмір набору початкових популяцій становить приблизно 300 екземплярів.

Отже виникає задача – на основі набору навчальних популяцій, який містить приблизно 300 хромосом-рішень, виявити малозначущі фактори. Для вирішення цієї задачі маємо застосовувати сучасні методи інтелектуального аналізу даних – ще однієї складової методології Data Farming. На сьогодні найбільш розвинений метод інтелектуального аналізу даних – це апарат нейромереж, можливість застосування якого надають відповідні відкриті бібліотеки, наприклад, Keras [17]. В стат-

ті запропоновано алгоритм виявлення малозначущих факторів за допомогою простої нейромережі, створеної з використанням Keras.

Опис алгоритму

1. Будується нейромережа, що реалізує регресію.

2. Формуються навчальні дані: вхідні та цільові тензори.

3. Нейромережа навчається на наявному наборі даних для того, щоб можна було передбачати значення цільової функції при довільних значеннях змінних з діапазону допустимих значень.

4. За допомогою навченої мережі оцінюється внесок кожного фактору до значення цільової функції двома способами.

Перший спосіб (додавання).

Знаходяться різниці між передбаченими значеннями цільової функції при заміні значення певного фактору з мінімально можливого на максимально можливе в наборі, де усі фактори мають мінімальні значення, та значенням передбаченої цільової функції для цього вихідного набору.

Другий спосіб (виключення).

Знаходяться різниці між передбаченими значеннями цільової функції при заміні значення певного фактору з максимально можливого на мінімально можливе в наборі, де усі фактори мають максимальні значення, та значенням передбаченої цільової функції для цього вихідного набору.

5. Отримані різниці вважаються оцінками значущості відповідних факторів та подаються в зручному для аналізу вигляді.

Побудова нейромережі за допомогою Keras

Keras – це відкрита високорівнева нейромережна бібліотека, написана мовою Python. Її основний автор та підтримувач – Франсуа Шоллє (фр. *François Chollet*). Keras призначена для зручного та швидкого конструювання моделей глибокого навчання та проведення різноманітних експериментів з ними. Вона не реалізує низькорівневі операції, такі як маніпуляції з тензорами та диференціювання, – для цього до неї підключаються різні спеціалізовані бібліотеки, такі як: TensorFlow, Theano, Microsoft Cognitive Toolkit, R.

Процес побудови нейромережі за допомогою Keras складається з наступних етапів:

1. Визначаються навчальні дані: вхідні та цільові тензори.

2. Визначаються шари мережі (модель), що відображають вхідні дані до цільових.

3. Налаштовується процес навчання вибором функції втрат, оптимізатора та деяких параметрів моніторингу.

4. Виконуються ітерації за навчальними даними шляхом виклику методу fit() моделі [18].

Нейромережа, що реалізує регресію, організована як простий стек повнозв'язних (Dense) шарів з операцією активації `relu: Dense(N, activation='relu')`.

Параметр N – кількість прихованих нейронів у шарі, функція активації relu має вигляд relu(x) = max(x, 0).

Повнозв'язні ("щільні", Dense) шари є одним із найважливіших компонентів нейронних мереж. Такий шар обробляє кожен елемент попереднього шару, виконуючи матричне перемноження цих елементів з їх вагами: `output = relu(dot(W, input) + b)`.

Потім отримані дані відправляються на наступний шар.

Прихований нейрон (hidden unit) – це вимір у просторі представлень шару. Наявність N прихованих нейронів означає, що вагова матриця W буде мати форму (input_dimension, N): скалярний добуток на W спроектує вхідні дані в *N*-мірний простір представлень (потім буде зроблено додавання з вектором зсувів *b* і виконана операція relu). Розмірність простору представлень можна інтерпретувати як «ступінь свободи нейронної мережі щодо внутрішніх представлень». Більша кількість прихованих нейронів (велика розмірність простору представлень) дозволяє мережі навчатися більш складних представлень, але при цьому збільшується обчислювальна вартість мережі, а та-

кож це може призвести до виявлення небажаних шаблонів (шаблонів, які можуть підвищити точність на навчальних даних, але не на контрольних).

Нейрони повнозв'язного шару з'єднані з усіма нейронами попереднього шару, що означає, що кожен нейрон повнозв'язного шару може взаємодіяти з будь-яким нейроном попереднього шару.

Використання повнозв'язних шарів у нейронних мережах надає можливість вивчення будь-якої функції, яка може бути представлена у вигляді математичної моделі. Нейрони повнозв'язних шарів можуть адаптуватися до будь-яких вхідних даних, навчаючись виділяти важливі ознаки та ігнорувати незначні.

Щодо стеку шарів Dense, потрібно прийняти два важливі архітектурні рішення:

1) яку кількість шарів використовувати;

2) скільки прихованих нейронів обрати для кожного шару.

Вибір оптимальних параметрів здійснюється шляхом експериментів.

Налаштування процесу навчання проводиться на етапі компіляції. При цьому задаються оптимізатор і функція втрат, які повинні використовуватися моделлю, а також всі метрики для моніторингу під час навчання.

Процес навчання полягає у передачі масивів Numpy з вхідними даними і відповідними цільовими даними до методу fit() моделі:

```
model.fit(input_tensor, target_tensor, batch_size=1, epochs=20)
```

Розмір пакета (batch_size) – це кількість зразків, оброблених до оновлення моделі. Діапазон його значень [1; *M*], де *M* – кількість зразків у навчальному наборі даних.

Розмір пакета – один з найважливіших гіперпараметрів, який являє собою кількість зразків, що використовуються в одному прямому та зворотному проході через мережу. Він безпосередньо впливає на точність та обчислювальну ефективність процесу навчання.

Мінімальне значення для розміру пакета дорівнює 1, що відповідає використанню кожного навчального екземпляра для виконання одного оновлення градієнта. Тут кількість оновлень градієнта або кількість навчальних кроків або кількість пакетів у одній епосі дорівнює розміру повного навчального набору даних.

Кількість епох (epochs) – це кількість повних проходів по навчальному набору даних. Як правило, оптимальна кількість епох становить від 1 до 20 і досягається, коли точність глибокого навчання перестає покращуватись.

Розробка моделі

Для проведення експериментів з метою визначення параметрів мережі використовувалися штучно створені набори даних. Зразок даних – це масив, що складається зі значень 100 факторів X_i і відповідного їм значення цільової функції Y. Змінні X_i приймають випадкові значення, рівномірно розподілені в діапазоні [0; 1] з кроком 1/30. Значення цільової функції Y обчислюється як сума значень змінних (факторів) з певними коефіцієнтами. Кількість зразків у наборі варіювалася від 200 до 300.

Оскільки кількість зразків даних порівняно невелика, спочатку використовувалася проста мережа з двома повнозв'язними проміжними шарами, що містять 64 приховані нейрони. Мережа закінчується одномірним шаром, який не має функції активації (лінійний шар). Це типова конфігурація для скалярної регресії (метою якої є передбачення одного значення на безперервній числовій прямій).

Мережа компілюється з функцією втрат mse – mean squared error (середньоквадратична похибка), що обчислює квадрат різниці між передбаченими та цільовими значеннями. Ця функція широ-

ко використовується у задачах регресії. Для моніторингу на етапі навчання використовується параметр mae – mean absolute error (середня абсолютна похибка).

```
from keras import models
from keras import layers
def build_model():
    model = models.Sequential()
    model.add(keras.Input(shape=(train_data.shape[1],)))
    model.add(layers.Dense(64, activation='relu'))
    model.add(layers.Dense(1))
    model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
    return model
```

Для оцінки якості мережі застосовувалася перехресна перевірка по k блоках [18]. Доступні дані діляться на k блоків (k = 4), створюються k ідентичних моделей, кожна навчається на k–1 блоках (тренувальні дані) з оцінкою по блоку, що залишився (тестові дані). За отриманими k оцінками обчислюється середнє значення, яке приймається як оцінка моделі.

Для візуалізації процесу навчання моделі зберігалася історія змін середньої абсолютної похибки (mae) перед кожною наступною епохою та створювалися діаграми залежності mae від кількості епох. Зразок діаграми показано на рис.1. Це дозволяє вибрати оптимальну кількість епох навчання, а також оцінити швидкість навчання в залежності від обраної архітектури мережі.

На графіку представлена залежність величини середньої абсолютної похибки від кількості епох навчання для набору даних, що містить 200 зразків, для випадку, коли перші 70 факторів входять до цільової функції з коефіцієнтами, рівними 1, а інші 30 – з коефіцієнтами 0.05.

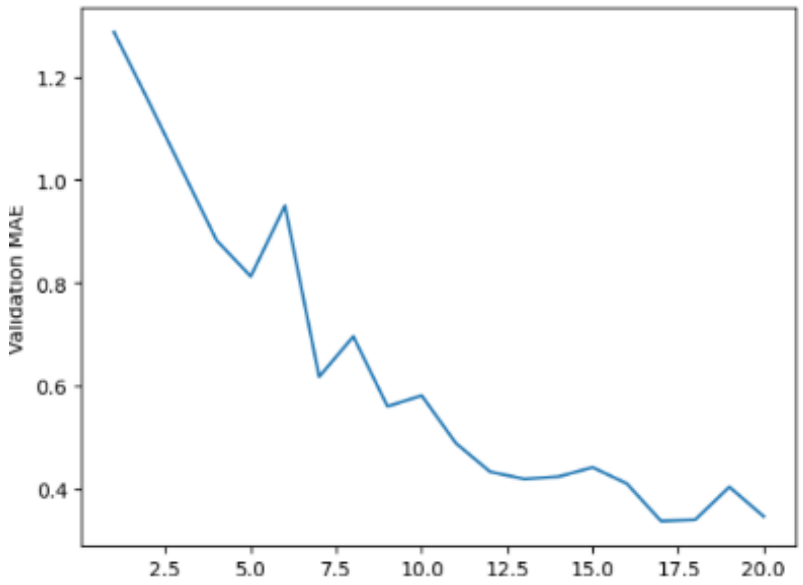

РИС. 1. Приклад діаграми залежності середньої абсолютної похибки (mae) від кількості епох навчання

Проводилися експерименти з наборами даних з різним співвідношенням значущих та малозначущих факторів, з різною кількістю зразків, а також з різними архітектурами мережі. Експерименти показали, що додавання додаткових шарів або додаткових прихованих нейронів не покращує якість передбачень нейромережі. Також було виявлено, що 15–20 епох цілком достатньо для отримання задовільних передбачень.

Після налаштування параметрів моделі її остаточна версія навчалася на всіх доступних даних і використовувалася для виявлення значущих та малозначущих факторів.

Використання моделі

Перша серія експериментів проводилася для наборів зразків, де значення були згенеровані за допомогою генератора випадкових чисел та зафіксовані. Навчання моделі відбувалося за 5–7 сек. Результати виводилися на діаграми.

На рис. 2 – 4 показано діаграми з оцінками значущості при різних співвідношеннях значущих та малозначущих атрибутів для набору, що складається з 300 зразків.

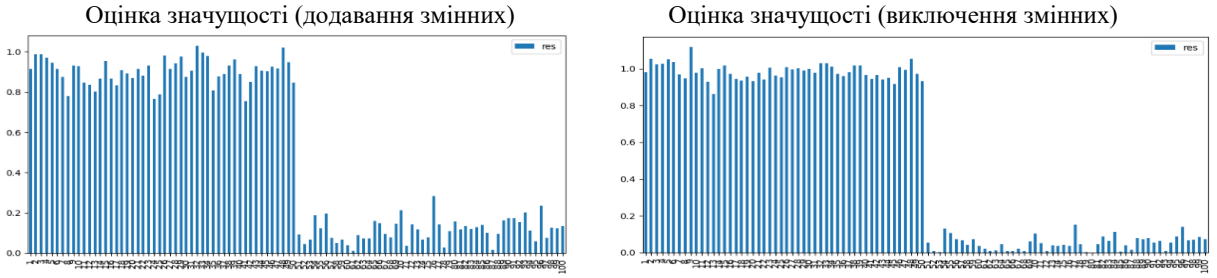

РИС. 2. Оцінки значущості для набору, де 50 перших факторів входять до цільової функції з коефіцієнтами 1, інші – з коефіцієнтами 0.05

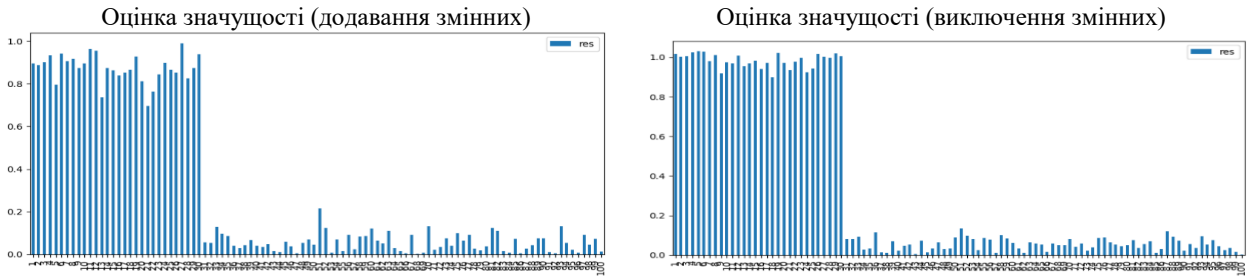

РИС. 3. Оцінки значущості для набору, де 30 перших факторів входять до цільової функції з коефіцієнтами 1, інші – з коефіцієнтами 0.05

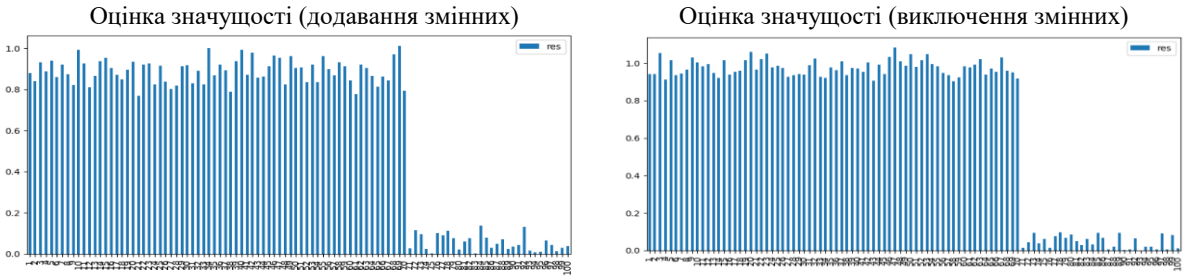

РИС. 4. Оцінки значущості для набору, де 70 перших факторів входять до цільової функції з коефіцієнтами 1, інші – з коефіцієнтами 0.05

Це дозволило виявити деякі закономірності.

Модель дозволяє впевнено визначити значущі та малозначущі фактори. Різні прогони на одному й тому ж наборі можуть давати дещо різні результати, оскільки початкові ваги нейромережі обираються випадковим чином. Але в цілому оцінка значущості способом виключення є більш

точною у порівнянні з оцінкою способом додавання. Наприклад, для розглянутого вище набору, що містить 70 значущих факторів та 30 малозначущих, було отримано такі оцінки (таблиця).

ТАБЛИЦЯ. Показники відхилення модельних оцінок від істинних значень в залежності від обраного способу оцінювання

Додавання	Виключення
Середнє значення абсолютного відхилення	
0.0991	0.0381
Максимальне абсолютне відхилення для значущих змінних	
0.2901	0.0968
Максимальне абсолютне відхилення для малозначущих змінних	
0.1008	0.0492

Набори початкових популяцій БГА

Друга серія експериментів проводилася з наборами початкових популяцій для дослідження БГА, який описано у вступній частині статті.

Досліджувалися також випадки з різним співвідношенням значущих та малозначущих факторів, та різним рівнем їх значущості.

Навчання мережі тривало близько 6 секунд. Алгоритм дозволив досить впевнено визначити значущі фактори для наборів, що містять від 8 до 10 популяцій.

На рис. 5 – 8 показано зразки відповідних діаграм для наборів, де коефіцієнти малозначущих факторів 0.1 та 0.2. Зменшення кількості популяцій негативно впливало на точність роботи алгоритму.

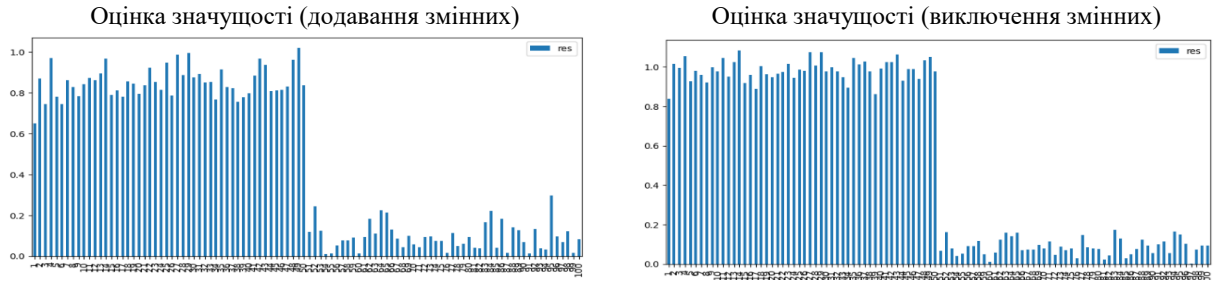

РИС. 5. Оцінки значущості для набору, що складається з 10 початкових популяцій, перші 50 факторів входять до цільової функції з коефіцієнтами 1, інші – з коефіцієнтами 0.1

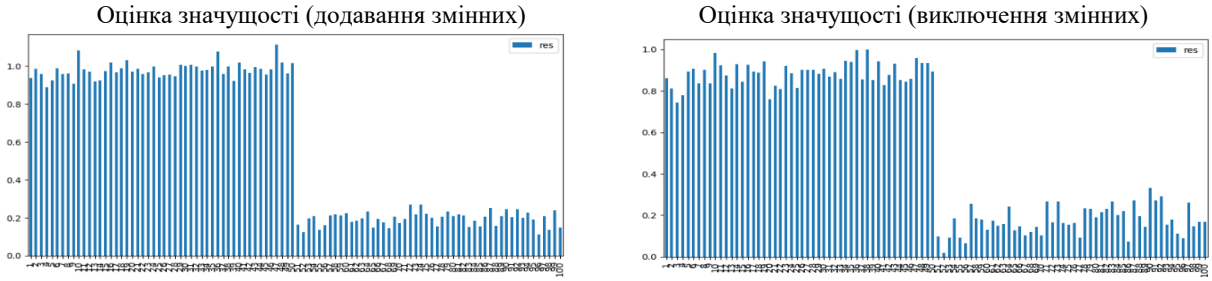

РИС. 6. Оцінки значущості для набору, що складається з 10 початкових популяцій, перші 50 факторів входять до цільової функції з коефіцієнтами 1, інші – з коефіцієнтами 0.2

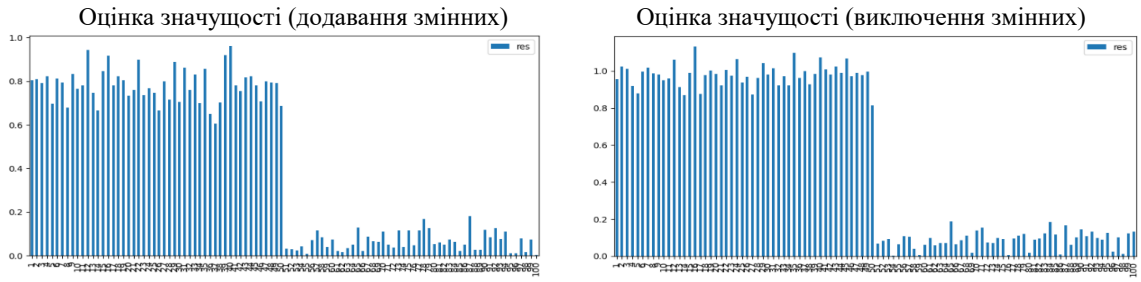

РИС. 7. Оцінки значущості для набору, що складається з 8 початкових популяцій, перші 50 факторів входять до цільової функції з коефіцієнтами 1, інші – з коефіцієнтами 0.1

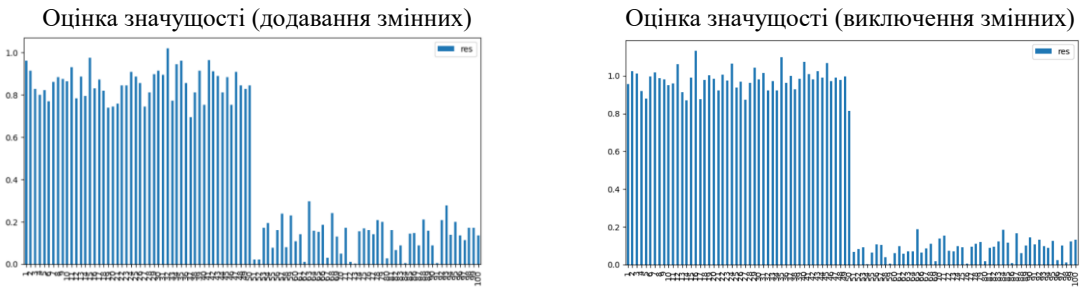

РИС. 8. Оцінки значущості для набору, що складається з 8 початкових популяцій, перші 50 факторів входять до цільової функції з коефіцієнтами 1, інші – з коефіцієнтами 0.2

В усіх випадках, так само, оцінка значущості способом виключення – більш точна у порівнянні з оцінкою способом додавання.

Висновки

Запропонований алгоритм на основі простої нейромережі дозволяє коректно та досить швидко визначати малозначущі фактори в наборі початкових популяцій для дослідження БГА, що містить від 8 до 10 популяцій (250 – 300 зразків). Оскільки початкові ваги нейромережі обираються випадковим чином, результати різних прогонів для одного й того ж набору можуть відрізнятися. Тому в загальному випадку оцінювання значущості факторів комп'ютерної моделі для отримання більш надійних результатів необхідно робити декілька прогонів.

Програмне забезпечення

Для обчислювальних експериментів використовувались:

- мова програмування: Python версії 3.11;
- середовище розробки: Jupyter notebook версії 7.0.4;
- бібліотеки: numpy, pandas, matplotlib, tensorflow, keras (ver.3.3.3).

Обчислення виконувались на ПК з процесором Intel Core i7 з ОП 8 Гб.

Авторські внески.

Пепеляєв В.А.: узагальнення, концептуалізація, методологія, формальний аналіз, написання – оригінальна чернетка, рецензування та редагування; Орєхова Н.А.: дослідження, методологія, формальний аналіз, візуалізація, написання – оригінальна чернетка; Лук'янов І.О.: програмне забезпечення, візуалізація.

Фінансування. Автори не отримували фінансування для проведення досліджень та написання статті.

Список літератури

1. Horne G.E., Meyer T.E. Data Farming: Discovering Surprise. *Prociding. of the Winter Simulation Conference*. 2005. P. 1082–1087.
2. SEED Center for Data Farming. http://harvest.nps.edu/ (звернення: 30.10.2024)
3. Barry Ph., Koehler M. Simulation in context: using Data Farming for decision Support. *Prociding. of the Winter Simulation Conference*. 2004. P. 814–819.
4. Horne G.E., Schwierz K.-P. Data Farming around the world overview. *Prociding. of the Winter Simulation Conference*. 2008. P. 1442–1447.
5. Choo C.S., Ng E.C., Ang D., Chua C.L. Data Farming: a brief history. *Prociding. of the Winter Simulation Conference*. 2008. P. 1448–1455.
6. Fu M. Optimization for Simulation: Theory and Practice. *INFORMS J. on Computing*. 2002. **14** (3). P. 192–215.
7. April J., Glover F., Kelly J.P., Laguna M. Practical introduction to simulation optimization. *Prociding. of the Winter Simulation Conference*. 2003. P. 71–78.
8. Пепеляев В.А. К вопросу об интеграции методов оптимизации и имитационного моделирования. *Теорія оптимальних рішень*. 2003. № 2. С. 51–61. http://dspace.nbuv.gov.ua/handle/123456789/84855
9. Pepelyaev V.A. Planning optimization-simulation experiments. *Cybernetics and Systems Analysis*. 2006. **42**, (6). P. 866–875. https://doi.org/10.1007/s10559-006-0126-z
10. Литвиненко Ф.А., Лукьянов И.О., Криковлюк Е.А. Особенности реализации параллельной версии многопопуляционного генетического алгоритма. *Компьютерная математика*. 2018. № 2. С. 21–29. http://dspace.nbuv.gov.ua/handle/123456789/161882
11. Литвиненко Ф.А., Лукьянов И.О., Криковлюк Е.А. О повышении эффективности параллельной версии многопопуляционного генетического алгоритма. *Теория оптимальных решений***.** 2019. С. 116–122. http://dspace.nbuv.gov.ua/handle/123456789/161683
12. Пепеляев В.А., Чёрный Ю.М. О возможностях применения генетических алгоритмов в оптимизационно-имитационных экспериментах. *Теорія оптимальних рішень*. 2019. № 18. С. 69–77. http://dspace.nbuv.gov.ua/handle/123456789/161681
13. Лукьянов И.О., Литвиненко Ф.А., Коваль В.П. О выборе размера начальной популяции для параллельной версии многопопуляционного генетического алгоритма. *IX Міжнародна школа-семінар «Теорія прийняття рішень»*. Україна. Ужгород. 15-20 квітня 2019. С. 95–96.
14. Лукьянов И.О., Литвиненко Ф.А., Козлюк Е.М. Об эффективном использовании начальной популяции генетического алгоритма. *XIX Міжнародна науково-технічна конференція «Проблеми інформатики та моделювання»*. Україна. Харків – Одеса. 11-16 вересня 2019. С. 52–53.
15. Лукьянов И.О. Об эффективности параллельного многопопуляционного генетического алгоритма для разного числа процессоров. *VII Международна научная конференция «Математическое моделирование, оптимизация и информационные технологии»***.** Кишинэу – Киев – Батуми. 15-19 ноября 2021.
16. Лук'янов І.О., Литвиненко Ф.А. Про вибір числа процесорів для паралельного багатопопуляційного генетичного алгоритму. *Кібернетика та комп'ютерні технології*. 2022. № 2. С. 31–37. https://doi.org/10.34229/2707-451X.22.2.3
17. About Keras 3. https://keras.io/about/ (звернення: 30.10.2024)
18. Chollet F. Deep learning with Python. Simon and Schuster. 2021.

Одержано 30.10.2024

Пепеляєв Володимир Анатолійович,
доктор фізико-математичних наук, старший науковий співробітник,
зав. відділом Інституту кібернетики імені В.М. Глушкова НАН України, Київ,
https://orcid.org/0009-0009-3169-1776

Орєхова Наталія Артурівна,
молодший науковий співробітник
Інституту кібернетики імені В.М. Глушкова НАН України, Київ,
naorexova@gmail.com

Лук'янов Ігор Олегович,
молодший науковий співробітник
Інституту кібернетики імені В.М. Глушкова НАН України, Київ.

УДК 519.711

В.А. Пепеляєв, Н.А. Орєхова *, І.О. Лук'янов

Оцінка значущості факторів комп'ютерної моделі за допомогою простої нейромережі

Інститут кібернетики імені В.М. Глушкова НАН України, Київ
* *Листування: naorexova@gmail.com*

Вступ. Існуючі засоби комп'ютерної техніки надають можливість розробляти моделі будь-якої складності. Ця обставина провокує розробників комп'ютерних моделей складних систем до зайвої деталізації. Серед досвідчених фахівців з комп'ютерного моделювання існує думка, що іноді на попередньому етапі розробки моделі кількість малозначущих факторів може досягати 80 %. Таке збільшення розмірності не тільки значно ускладнює здійснення комп'ютерних експериментів, але й може мати суттєвий вплив на розуміння взаємодії важливих факторів, які визначають основу і суть функціонування складної системи. Тому не менш важливим для подальшого модельного дослідження, а особливо оптимізації складної системи, є визначення малозначущих факторів.

Мета роботи. Розробити алгоритм визначення малозначущих факторів за наявності набору навчальних даних, в якому кількість зразків даних порівняно невелика й перевищує кількість факторів всього в 2-3 рази. Для цього було використано модель нейромережі, що реалізує регресію, створену за допомогою бібліотеки Keras. Для проведення експериментів з метою визначення параметрів мережі (кількості шарів, кількості прихованих нейронів в шарі, а також кількості епох навчання) використовувалися штучно створені набори даних.

Результати. Отримана модель нейромережі продемонструвала ефективну роботу на тестових наборах даних. Потім модель використовувалася для визначення значущості факторів у наборах початкових популяцій для дослідження багатопопуляційного генетичного алгоритму (БГА).

Висновки. Запропонований алгоритм на основі простої нейромережі дозволяє коректно та досить швидко визначати малозначущі фактори в наборі початкових популяцій для дослідження БГА, що містить від 8 до 10 популяцій (250 – 300 зразків). Оскільки початкові ваги нейромережі обираються випадковим чином, результати різних прогонів для одного й того ж набору дещо відрізняються. Тому в загальному випадку оцінювання значущості факторів комп'ютерної моделі для отримання більш надійних результатів необхідно робити декілька прогонів.

Ключові слова: набір початкових популяцій, значущість факторів, нейромережа, епохи навчання.

UDC 519.711

Volodymyr Pepelyaev, Nataliia Oriekhova *, Ihor Lukyanov

Estimating the Significance of Computer Model Factors Based on a Simple Neural Network

V.M. Glushkov Institute of Cybernetics of the NAS of Ukraine, Kyiv
* *Correspondence: naorexova@gmail.com*

Introduction. The existing means of computer technology provide an opportunity to develop models of any complexity. This circumstance provokes developers of computer models of complex systems to excessive detailing. Among experienced specialists in computer modeling, there is an opinion that sometimes at the preliminary stage of model development, the number of insignificant factors can reach 80%. Such an increase in dimensionality not only significantly complicates the implementation of computer experiments, but can also have a significant impact on the understanding of the interaction of important factors that determine the basis

and essence of the functioning of a complex system. Therefore, it is no less important for further model research, and especially for the optimization of a complex system, to determine insignificant factors.

The purpose of the work is to develop an algorithm for determining insignificant factors in the presence of a set of training data, in which the number of data samples is relatively small and exceeds the number of factors by only 2-3 times. For this, a neural network model implementing regression created using the Keras library was used. Artificially created datasets were used to conduct experiments to determine network parameters (number of layers, number of hidden neurons in a layer, as well as the number of learning epochs).

The results. The resulting neural network model demonstrated effective performance on test data sets. The model was then used to determine the significance of factors in sets of initial populations for a multipopulational genetic algorithm (MGA) study.

Conclusions. The proposed algorithm based on a simple neural network allows to correctly and quickly determine insignificant factors in a set of initial populations for the study of MGA, containing from 8 to 10 populations (250 - 300 samples). Since the initial weights of the neural network are chosen randomly, the results of different runs on the same set of data are slightly different. Therefore, in the general case of evaluating the significance of factors of a computer model, several runs must be made to obtain more reliable results.

Keywords: set of initial populations, significance of factors, neural network, learning epochs.