

## **ПАРАЛЕЛЬНИЙ ГРУПОВИЙ АЛГОРИТМ НАВЧАННЯ БАГАТОШАРОВОГО ПЕРСЕПТРОНУ З ДВОМА СХОВАНИМИ ШАРАМИ НЕЙРОНІВ**

*В даній статті представлено паралельний груповий алгоритм зворотного поширення помилки для навчання багатошарового перцептрону з двома схованими шарами нейронів та результати досліджень його ефективності розпаралелення на багатоядерній високопродуктивній обчислювальній системі. Модель багатошарового перцептрону та груповий алгоритм навчання описані теоретично. В статті представлено блок-схему розробленого паралельного групового алгоритму навчання. Результати експериментальних досліджень показали високу ефективність розпаралелення запропонованого алгоритму навчання для задачі класифікації великої розмірності на багатоядерній обчислювальній системі з 48-ма ядрами з використанням технології MPI.*

*Ключові слова:* Паралельне групове навчання, багатошаровий перцептрон, ефективність розпаралелення, класифікація даних.

VOLODYMYR TURCHENKO, VITALIY DOROSH, IGOR PALIY  
Ternopil National Economic University

## **PARALLEL BATCH PATTERN TRAINING ALGORITHM FOR MULTILAYER PERCEPTRON WITH TWO HIDDEN LAYERS OF NEURONS**

*The development of parallel batch pattern back propagation training algorithm of multilayer perceptron with two hidden layers and its parallelization efficiency research on many-core high performance computing system are presented in this paper. The model of multilayer perceptron and the batch pattern training algorithm are theoretically described. The algorithmic description of the parallel batch pattern training method is presented. Our results show high parallelization efficiency of the developed training algorithm on large scale data classification task on many-core parallel computing system with 48 CPUs using MPI technology.*

*Keywords:* Parallel batch pattern training, multilayer perceptron, parallelization efficiency, data classification.

### **ВСТУП**

Штучні нейронні мережі (НМ) є чудовим механізмом для моделювання складних нелінійних систем. Вони є дуже доброю альтернативою традиційним методам рішення складних задач в багатьох сферах включаючи обробку зображень, розпізнавання образів, робототехніку, оптимізацію, моделювання процесів життєдіяльності та інших [1]. В загальному випадку, НМ складається з двох і більше шарів елементарних нейронів з'єднаних ваговими коефіцієнтами. Кожен нейрон поточного шару отримує інформацію від нейронів попереднього шару, виконує операцію зваженої суми, використовує її як аргумент для обчислення функції активації та передає результат до нейронів наступного шару. Хоча обчислення в кожному нейроні та НМ загалом є дуже прості, загальне обчислювальне навантаження може бути значним, особливо на етапі навчання НМ (години та дні). Це, без сумніву, є однією з головних перепон до ефективного використання НМ для вирішення прикладних задач. Використання високопродуктивних паралельних комп'ютерів, суперкомп'ютерів, кластерів та обчислювальних ГРІД-систем загального призначення для прискорення етапу навчання НМ є шляхом усунення цієї перепони. Тому розробка паралельних методів навчання НМ та дослідження їх ефективності розпаралелення на таких класах паралельних комп'ютерних систем все ще залишається актуальною науковою задачею.

Беручи до уваги паралельну природу штучних НМ, багато дослідників сконцентрували свою увагу на їх розпаралеленні. Автори [2] досліджують паралельне навчання багатошарового перцептрону (БШП) на багатопроцесорному комп'ютері, кластері та обчислювальній ГРІД-системі, використовуючи пакет MPI (Message Passing Interface). Дослідження паралельного алгоритму навчання простої рекурентної нейронної мережі Елмана, заснованого на розширеному фільтрі Калмана на багатоядерних системах і графічних процесорах представлено у [3]. Автори статті [4] представили розробку паралельного алгоритму навчання повно-зв'язної рекурентної нейронної мережі, що базується на правилі навчання «нагорода-штраф». В рамках розробки паралельної ГРІД-базованої бібліотеки для навчання нейронних мереж [5], був розроблений паралельний алгоритм зворотного поширення помилки для навчання БШП з одним схованим шаром нейронів [6], рекурентної НМ [7], рециркуляційної НМ [8] та НМ з радіально-базисною функцією активації [9]. Результати цих досліджень показали хорошу ефективність розпаралелення на різних архітектурах високопродуктивних комп'ютерних систем.

Однак аналіз останніх досліджень показав, що розпаралелення БШП з двома схованими шарами нейронів не є ще досліджено належним чином. Наприклад, автори [2] розпаралелюють архітектуру БШП 16-10-10-1 (16 нейронів у вхідному шарі, два схованих шари по 10 нейронів у кожному та один вихідний нейрон), що обробляє велику кількість навчальних векторів (близько 20000) з Великого Андронного Коллайдера. Їх реалізація відносно «малої» архітектури НМ з 270 внутрішніми налаштовуваними зв'язками (загальна кількість вагових коефіцієнтів та порогів НМ) не забезпечує прискорення розпаралелення через значні втрати часу на обмін даними між паралельними процесорами. На нашу думку, ці втрати зумовлені тим, що «комунікаційна частина» алгоритму не оптимізована і містить принаймні три фізичних виклики

функції передачі повідомлення бібліотеки MPI між гілками паралельної програми.

Згідно з теоремою універсального апроксиматора [10, 11], БШП з двома схованими шарами забезпечує кращий контроль процесу апроксимації, ніж БШП з одним схованим шаром [1]. Також БШП з двома схованими шарами дозволяє обробляти "глобальні особливості", тобто нейрони першого схованого шару збирають "локальні особливості" від вхідних даних, а нейрони другого схованого шару узагальнюють отримані результати першого шару, що забезпечує відображення даних більш високого рівня. Це особливо актуально для вирішення завдань класифікації, розпізнавання об'єктів та комп'ютерного бачення. Однак обчислювальний час для такого роду завдань надзвичайно великий. Наприклад, навчання мережі з 500 нейронів у першому схованому шарі та 300 нейронів у другому схованому шарі на 60000 навчальних векторах бази даних MNIST [12] є дуже повільним; тільки 59 епох попереднього навчання обох схованих шарів як обмеженої машини Больцмана (Restricted Boltzmann Machine) тривало близько тижня [13]. Тому дослідження ефективності паралельного алгоритму навчання БШП з двома схованими шарами нейронів є актуальною проблемою.

Враховуючи, що груповий алгоритм навчання за правилом зворотного поширення помилки показав хорошу ефективність розпаралелення на кількох архітектурах НМ, метою даної статті є розробка цього паралельного алгоритму для БШП з двома схованими шарами та дослідження його ефективності розпаралелення на задачі класифікації даних великої розмірності на багатоядерних високопродуктивних обчислювальних системах.

### ОСНОВНА ЧАСТИНА

Груповий алгоритм навчання змінює вагові коефіцієнти та пороги нейронів в кінці кожної навчальної епохи, тобто після обробки всіх векторів навчання з навчальної вибірки, замість зміни вагових коефіцієнтів і порогів після послідовної обробки кожного навчального вектору [6]. Вихідні значення БШП з двома схованими шарами (Рис 1.) описується виразами [1]:

$$y_{3g} = F_3(S_{3g}), S_{3g} = \sum_{k=1}^K y_{2k} \cdot w_{3kg} - T_{3g} \quad (1)$$

$$y_{2k} = F_2(S_{2k}), S_{2k} = \sum_{j=1}^N y_{1j} \cdot w_{2jk} - T_{2k}, \quad (2)$$

$$y_{1j} = F_1(S_{1j}), S_{1j} = \sum_{i=1}^M x_i \cdot w_{1ij} - T_{1j}, \quad (3)$$

де  $F_3, F_2$  і  $F_1$  - сигмоїдні функції активації  $F(x) = 1/(1 + e^{-x})$ , що використовуються для нейронів вихідного і двох схованих шарів відповідно,  $S_{3g}, S_{2k}$  і  $S_{1j}$  - зважені суми нейронів вихідного і двох схованих шарів відповідно,  $K, N$  і  $M$  - кількість нейронів двох схованих і вхідного шарів відповідно,  $y_{2k}$  - виходи нейронів другого схованого шару,  $w_{3kg}$  - вагові коефіцієнти від  $k$ -го нейрону другого схованого шару до  $g$ -го вихідного нейрону,  $T_{3g}$  - порогове значення  $g$ -го вихідного нейрону,  $y_{1j}$  - виходи нейронів першого схованого шару,  $w_{2jk}$  - вагові коефіцієнти від  $j$ -го нейрону першого схованого шару до  $k$ -го нейрону другого схованого шару,  $T_{2k}$  - пороги  $k$ -го нейрону другого схованого шару,  $x_i$  - вхідні значення,  $w_{1ij}$  - вагові коефіцієнти від  $i$ -го вхідного нейрону до  $j$ -го нейрону першого схованого шару і  $T_{1j}$  - пороги нейронів першого схованого шару.

Груповий алгоритм навчання за правилом зворотного поширення помилки для цієї архітектури БШП складається з наступних кроків [14]:

1. Встановити необхідну середньоквадратичну помилку  $SSE = E_{\min}$  та кількість навчальних епох  $t$ ;

2. Ініціалізувати випадковим чином вагові коефіцієнти та пороги нейронів значеннями в діапазоні  $(0 \dots 0,5)$  [14];

3. Для навчання вектору  $pt$ :

3.1. Обчислити значення вихідних нейронів

$$y_{3g}^{pt}(t) \text{ згідно з виразами (1), (2) і (3);}$$

3.2. Обчислити помилку вихідних нейронів

$$\gamma_{3g}^{pt}(t) = y_{3g}^{pt}(t) - d_g^{pt}(t), \text{ де } y_{3g}^{pt}(t) - \text{обчислене значення } g\text{-го вихідного нейрону БШП, а } d_g^{pt}(t) - \text{бажане вихідне значення БШП;}$$

3.3. Обчислити (і) помилки нейронів другого схованого шару  $\gamma_{2k}^{pt}(t) = \gamma_{3g}^{pt}(t) \cdot w_{3kg}(t) \cdot F'_3(S_{3g}^{pt}(t))$ , де

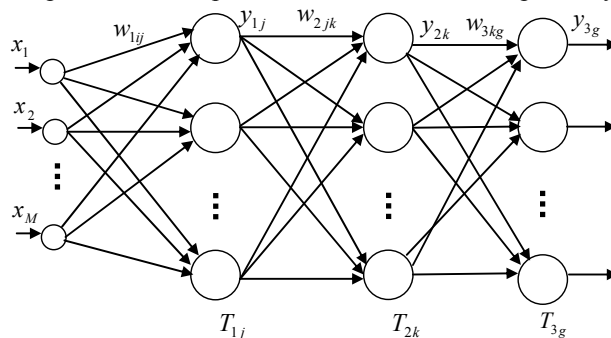


Рис. 1. Структура БШП з двома схованими шарами

$S_{3g}^{pt}(t)$  - зважена сума  $g$ -го вихідного нейрону та (ii) помилки нейронів першого схованого шару

$\gamma_{1j}^{pt}(t) = \sum_{k=1}^K \gamma_{2k}^{pt}(t) \cdot w_{2jk}(t) \cdot F'_2(S_{2k}^{pt}(t))$ , де  $S_{2k}^{pt}(t)$  - зважена сума  $k$ -нейрону другого схованого шару;

3.4. Розрахувати зміни вагових коефіцієнтів та порогів усіх нейронів і додати результат до значення, отриманого на попередньому векторі навчання:

$$s\Delta w_{3g} = s\Delta w_{3g} + \gamma_{3g}^{pt}(t) \cdot F'_3(S_{3g}^{pt}(t)) \cdot y_{2k}^{pt}(t), \quad s\Delta T_{3j} = s\Delta T_{3j} + \gamma_{3g}^{pt}(t) \cdot F'_3(S_{3g}^{pt}(t)), \quad (4)$$

$$s\Delta w_{2jk} = s\Delta w_{2jk} + \gamma_{2k}^{pt}(t) \cdot F'_2(S_{2k}^{pt}(t)) \cdot y_{1j}^{pt}(t), \quad s\Delta T_{2j} = s\Delta T_{2j} + \gamma_{2k}^{pt}(t) \cdot F'_2(S_{2k}^{pt}(t)), \quad (5)$$

$$s\Delta w_{1ij} = s\Delta w_{1ij} + \gamma_{1j}^{pt}(t) \cdot F'_1(S_{1j}^{pt}(t)) \cdot x_i^{pt}(t), \quad s\Delta T_{1j} = s\Delta T_{1j} + \gamma_{1j}^{pt}(t) \cdot F'_1(S_{1j}^{pt}(t)), \quad (6)$$

3.5. Визначити середньоквадратичну помилку навчання

$$E^{pt}(t) = \frac{1}{2} (y_{3g}^{pt}(t) - d_g^{pt}(t))^2; \quad (7)$$

4. Повторити крок 3 для кожного навчального вектору  $pt$ , де  $pt \in \{1, \dots, PT\}$ , де  $PT$  - розмір навчальної вибірки;

5. Оновити вагові коефіцієнти та порогові нейронів використовуючи вирази  $w_{3kg}(PT) = w_{3kg}(0) - \alpha_3(t) \cdot s\Delta w_{3kg}$ ,  $T_{3g}(PT) = T_{3g}(0) + \alpha_3(t) \cdot s\Delta T_{3g}$ ,  $w_{2jk}(PT) = w_{2jk}(0) - \alpha_2(t) \cdot s\Delta w_{2jk}$ ,  $T_{2k}(PT) = T_{2k}(0) + \alpha_2(t) \cdot s\Delta T_{2k}$ ,  $w_{1ij}(PT) = w_{1ij}(0) - \alpha_1(t) \cdot s\Delta w_{1ij}$ ,  $T_{1j}(PT) = T_{1j}(0) + \alpha_1(t) \cdot s\Delta T_{1j}$ ,

де  $\alpha_3(t)$ ,  $\alpha_2(t)$  і  $\alpha_1(t)$  - швидкості навчання для вихідного та двох схованих шарів нейронів відповідно;

6. Розрахувати загальну середньоквадратичну помилку  $E(t)$  для навчальної епохи  $t$  використовуючи вираз

$$E(t) = \sum_{pt=1}^{PT} E^{pt}(t);$$

7. Якщо  $E(t)$  більше ніж очікувана помилка навчання, тоді збільшити число навчальних епох до  $t+1$  і перейти до кроку 3, інакше припинити процес навчання.

Як і у випадку БШП з одним схованим шаром [6], послідовне виконання кроків 3.1-3.5 для всіх навчальних векторів можна розпаралелити, тому часткові суми  $s\Delta w_{3kg}$ ,  $s\Delta T_{3g}$ ,  $s\Delta w_{2jk}$ ,  $s\Delta T_{2k}$ ,  $s\Delta w_{1ij}$  і  $s\Delta T_{1j}$  є незалежними одна від одної. Таким чином, всю обчислювальну роботу можна розділити між процесорами *Master* (цей процесор здійснює призначення функцій та виконання обчислень) та *Workers* (ці процесори здійснюють тільки виконання обчислень).

Алгоритми для процесорів *Master* і *Workers* зображені на рис. 2. Свою роботу *Master* починає з визначення (i) кількості векторів  $PT$  у навчальній вибірці та (ii) кількості процесорів  $p$ , на яких повинно здійснюватися виконання паралельного алгоритму. *Master* ділить всі вектори на рівні частини відповідно до кількості наявних процесорів *Workers* і призначає одну частину векторів собі. Тоді *Master* відправляє до кожного з процесорів *Workers* відповідну кількість векторів для навчання.

Кожен *Worker* виконує наступні дії для кожного навчального вектору  $pt$ :

1. Обчислює кроки 3.1-3.5 та 4 тільки для своєї кількості навчальних векторів. Значення часткових сум вагових коефіцієнтів  $s\Delta w_{3kg}$ ,  $s\Delta w_{2jk}$ ,  $s\Delta w_{1ij}$  та порогів  $s\Delta T_{3g}$ ,  $s\Delta T_{2k}$ ,  $s\Delta T_{1j}$  розраховуються на цьому кроці;

2. Обчислює часткову середньоквадратичну помилку SSE для встановленої кількості навчальних векторів.

В розробленому алгоритмі, після обробки всіх призначених навчальних векторів, виконується тільки одна операція колективного збору часткових результатів обчислень `MPI_Allreduce`, яка автоматично забезпечує сумування та синхронізацію з іншими процесорами [15]. Після того об'єднані значення  $s\Delta w_{3kg}$ ,  $s\Delta T_{3g}$ ,  $s\Delta w_{2jk}$ ,  $s\Delta T_{2k}$ ,  $s\Delta w_{1ij}$  та  $s\Delta T_{1j}$  направляються назад до всіх процесорів, що працюють паралельно, і розміщуються в локальній пам'яті кожного процесора. Кожен процесор використовує ці значення для оновлення вагових коефіцієнтів і порогів відповідно до кроку 5 вищезазначеного алгоритму, щоб використати їх в наступній ітерації навчання. Сумарне значення середньоквадратичної помилки навчання  $E(t)$  також отримується в результаті операції збору даних від паралельних процесорів, і процесор *Master* вирішує, продовжувати навчання чи ні.

Програмне забезпечення паралельного алгоритму розроблене на мові програмування C

використовуючи стандартні функції MPI. Паралельна частина алгоритму починається з виклику функції  $MPI\_Init()$ . Функція  $MPI\_Allreduce()$  здійснює збір часткових значень вагових коефіцієнтів  $s\Delta w_{3k}$ ,  $s\Delta w_{2jk}$ ,  $s\Delta w_{1ij}$  та порогів  $s\Delta T_3$ ,  $s\Delta T_{2k}$ ,  $s\Delta T_{1j}$ , сумує їх і відправляє назад до всіх процесорів в групі. Так як

вагові коефіцієнти і пороги фізично розташовані в різних змінних/матрицях програми, то доцільно здійснити попереднє кодування всіх даних в одному повідомленні перед його відправкою та зворотнє декодування даних у відповідні матриці після прийому повідомлення для того, щоб забезпечити тільки один фізичний виклик функції  $MPI\_Allreduce()$  в комунікаційній секції алгоритму. Функція  $MPI\_Finalize()$  завершує паралельну частину алгоритму навчання.

Багатоядерна високопродуктивна обчислювальна система *Remus*, розташована в лабораторії інноваційних обчислень Університету штату Теннессі (США), була використана для проведення паралельних обчислень. *Remus* складається з двох материнських плат RD890 (чіпсет AMD 890FX) пов'язаних між собою комунікаційним зв'язком AMD Hyper Transport. Кожна материнська плата містить два дванадцяти-ядерних процесори AMD Opteron 6180 SE з тактовою частотою 2500 МГц і 132 ГБ локальної оперативної пам'яті. Таким чином, загальна кількість обчислювальних ядер - 48. Кожен процесор має кеш L2 розміром 12x512 Кб, а розмір кеш-пам'яті L3 становить 2x6 Мб. Експерименти проведені з використанням бібліотеки Open MPI 1.6.3 [16].

Аналогічно роботі [13], база даних рукописних цифр MNIST, що містить 60 000 навчальних зображень і 10000 тестових зображень, використана для дослідження ефективності розпаралелення задачі класифікації даних. Розмір вхідних зображень в базі даних MNIST дорівнює 784 елементів; тому кількість вхідних нейронів БШП буде дорівнювати 784. Для того, щоб оцінити ефективність розпаралелення на різних розмірах обчислювальної задачі, кількість нейронів першого схованого шару змінювалась значеннями 80, 160, 240, 320 і 400. Аналогічно, кількість нейронів другого схованого шару змінювалась значеннями 50, 100, 150, 200 і 250. В базі даних MNIST є десять класів, цифри 0 ... 9, таким чином, використаний БШП містив 10 вихідних нейронів. Навчання здійснювалось шляхом постійного зростання кількості тренувальних векторів, з 10000 векторів у найменшому сценарії до 50000 векторів за найбільшим сценарієм. Таким чином, досліджено наступні сценарії розпаралелення: 784-80-50-10 / 10000 векторів, 784-160-100-10 / 20000 векторів, 784-240-150-10 / 30000 векторів, 784-320-200-10 / 40000 векторів та і 784-400-250-10 / 50000 векторів. Останній сценарій трохи менший від сценарію 784-500-300-10 / 60000, дослідженого в статті [13].

Результати досліджень в [17] показали, що ефективність розпаралелення групового паралельного алгоритму навчання не залежить від кількості тренувальних епох, так як вагові коефіцієнти та пороги нейронів об'єднуються в кінці кожної епохи. Тому, беручи до уваги можливий значний обчислювальний час всього експерименту, доцільно дослідити ефективність розпаралелення паралельного алгоритму навчання тільки на ста навчальних епохах. Кроки навчання були вибрані постійними і дорівнювали  $\alpha_1(t) = 0.6$ ,  $\alpha_2(t) = 0.6$  та  $\alpha_3(t) = 0.6$ . Вирази  $S = T_s / T_p$  і  $E = S / p \times 100\%$  використані для обчислення прискорення та ефективності розпаралелення, в яких  $T_s$  - це час послідовного виконання послідовної версії програми на одному процесорі,  $T_p$  - це час виконання паралельної версії тієї ж програми на  $p$  процесорах паралельної системи.

Графіки ефективності розпаралелення групового алгоритму навчання на задачі класифікації великої розмірності за допомогою БШП з двома схованими шарами нейронів для ста навчальних епох та 5-ти досліджуваних сценаріїв представлені на Рис. 3: ефективність розпаралелення становить 96-98% на 48-ми ядрах для трьох великих сценаріїв розпаралелення та 72-78% на 48-ми ядрах для двох менших сценаріїв

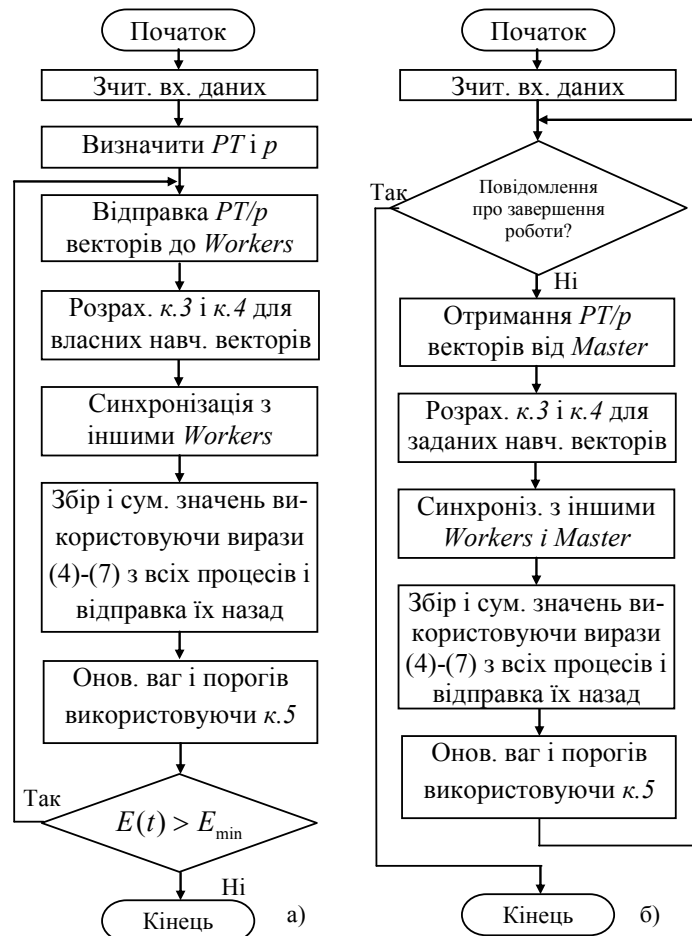


Рис. 2. Алгоритми процесорів Master (а) і Workers (б)

розпаралелення.

Таким чином, з урахуванням отриманих результатів ефективності розпаралелення, задача класифікації даних великої розмірності, якій може знадобитися, наприклад,  $10^4$  навчальних епох для БШП 784-400-250-1 з двома схованими шарами нейронів на 50000 навчальних векторів бази даних MNIST обчислюватиметься приблизно 21 день шляхом виконання послідовної програми і тільки 10 з половиною годин на 48-ми ядрах багатоядерної високопродуктивної обчислювальної системи.

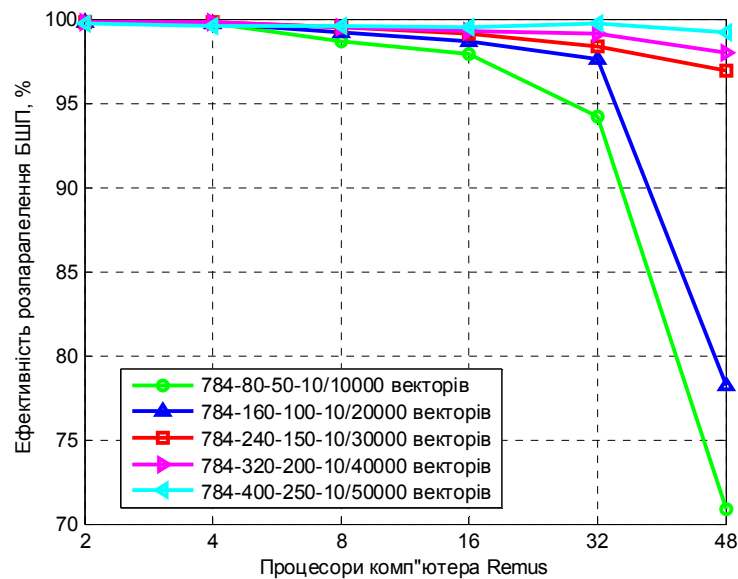


Рис. 3. Ефективність розпаралелення алгоритму на багатоядерній системі

## ВИСНОВКИ

Розробка паралельного групового алгоритму навчання на основі зворотного поширення помилки для багатощарового персептрону з двома схованими шарами нейронів та дослідження його ефективності розпаралелення для задачі класифікації даних великої розмірності на багатоядерній високопродуктивній обчислювальній системі представлені в цій статті. Модель багатощарового персептрону та груповий алгоритм навчання описані теоретично. Представлений алгоритмічний опис паралельного групового алгоритму навчання. Результати експериментальних досліджень показали високу ефективність розпаралелення розробленого алгоритму на багатоядерній системі з 48-ма процесорами: ефективність розпаралелення становить (i) 96-98% для трьох великих сценаріїв з 240, 320 і 400 нейронами у першому прихованому шарі та 150, 200 і 250 нейронами в другому прихованому шарі та 30000, 40000 і 50000 навчальних векторів відповідно та (ii) 72-78% для двох менших сценаріїв з 80 і 160 нейронами у першому схованому шарі та 50 і 100 нейронами у другому схованому шарі з 10000 і 20000 навчальними векторами відповідно. Майбутнім напрямком нашого дослідження буде розробка паралельних алгоритмів навчання для інших багатощарових архітектур нейронних мереж, зокрема згорткової нейронної мережі.

## Література

1. Haykin S. Neural Networks and Learning Machines / Simon Haykin // New Jersey: Prentice Hall, 2008. – 936 p.
2. De Llano R. M. Study of Neural Net Training Methods in Parallel and Distributed Architectures / R. M. De Llano, J. L. Bosque // Future Generation Computer Systems. – 2010. – No 26. – P. 183–190.
3. Cernansky M. Training Recurrent Neural Network Using Multistream Extended Kalman Filter on Multicore Processor and CUDA Enabled GPU / M. Cernansky // Heidelberg: Springer, Lecture Notes in Computer Science. – 2009. – Vol. 5768 (I). – P. 381–390.
4. Lotric U. Parallel Implementations of Recurrent Neural Network Learning / U. Lotric, A. Dobnikar // Heidelberg: Springer, Lecture Notes in Computer Science. – 2009. – Vol. 5495. – P. 99–108.
5. Parallel Grid-aware Library for Neural Network Training [Electronic resource] – Access mode: <http://uweb.deis.unical.it/turchenko/research-projects/pagalinnet/>
6. Turchenko V. Scalability of Enhanced Parallel Batch Pattern BP Training Algorithm on General-Purpose Supercomputers / V. Turchenko, L. Grandinetti // Advances in Intelligent and Soft-Computing. – 2010. – Vol. 79. – P. 518–526.
7. Turchenko V. Parallel Batch Pattern BP Training Algorithm of Recurrent Neural Network / V. Turchenko, L. Grandinetti // 14th IEEE International Conference on Intelligent Engineering Systems. – Las Palmas of Gran Canaria (Spain). – 2010. – P. 25–30.
8. Turchenko V. Efficient Parallelization of Batch Pattern Training Algorithm on Many-core and Cluster Architectures / V. Turchenko, G. Bosilca, A. Bouteiller, J. Dongarra. // 7th IEEE International Conference on

- Intelligent Data Acquisition and Advanced Computing Systems. – Berlin (Germany). – 2013. – P. 692–698.
9. Turchenko V. Training Algorithm for Radial Basis Function Neural Network / V. Turchenko, V. Golovko, A. Sachenko // 7th International Conference on Neural Networks and Artificial Intelligence. – Minsk (Belarus). – 2012. – P. 47–51.
  10. Funahashi K. On the Approximate Realization of Continuous Mappings by Neural Network / K. Funahashi // Neural Networks. – 1989. – No 2. – P. 183–192.
  11. Hornik K. Multilayer Feedforward Networks are Universal Approximators / K. Hornik, M. Stinchcombe, H. White // Neural Networks. – 1989. – No 2. – P. 359–366.
  12. The MNIST Database of Handwritten Digits [Electronic resource] – Access mode: <http://yann.lecun.com/exdb/mnist/>
  13. Hinton G. E. A Fast Learning Algorithm for Deep Belief Nets / G. E. Hinton, S. Osindero, Y. The // Neural Computation. – 2006. – No 18. – P. 1527–1554.
  14. Golovko V. Neural Networks: Training, Models and Applications / V. Golovko, A. Galushkin // Moscow: Radiotekhnika, 2001, (in Russian).
  15. Turchenko V. Improvement of Parallelization Efficiency of Batch Pattern BP Training Algorithm Using Open MPI / V. Turchenko, L. Grandinetti, G. Bosilca, J. Dongarra // Procedia Computer Science. – 2010. – No 1, Issue 1. – P. 525–533.
  16. Open MPI: Open Source High Performance Computing [Electronic resource] – Access mode: <http://www.open-mpi.org/>
  17. Turchenko V. Scalability of Parallel Batch Pattern Neural Network Training Algorithm / V. Turchenko // Artificial Intelligence, Journal of Institute of Artificial Intelligence of National Academy of Sciences of Ukraine. – Donetsk. – 2009. – No 2. – P. 144–150.

#### References

1. Haykin S. Neural Networks and Learning Machines / Simon Haykin // New Jersey: Prentice Hall, 2008. – 936 p.
2. De Llano R. M. Study of Neural Net Training Methods in Parallel and Distributed Architectures / R. M. De Llano, J. L. Bosque // Future Generation Computer Systems. – 2010. – No 26. – P. 183–190.
3. Cernansky M. Training Recurrent Neural Network Using Multistream Extended Kalman Filter on Multicore Processor and CUDA Enabled GPU / M. Cernansky // Heidelberg: Springer, Lecture Notes in Computer Science. – 2009. – Vol. 5768 (I). – P. 381–390.
4. Lotric U. Parallel Implementations of Recurrent Neural Network Learning / U. Lotric, A. Dobnikar // Heidelberg: Springer, Lecture Notes in Computer Science. – 2009. – Vol. 5495. – P. 99–108.
5. Parallel Grid-aware Library for Neural Network Training [Electronic resource] – Access mode: <http://uweb.deis.unical.it/turchenko/research-projects/pagalinnet/>
6. Turchenko V. Scalability of Enhanced Parallel Batch Pattern BP Training Algorithm on General-Purpose Supercomputers / V. Turchenko, L. Grandinetti // Advances in Intelligent and Soft-Computing. – 2010. – Vol. 79. – P. 518–526.
7. Turchenko V. Parallel Batch Pattern BP Training Algorithm of Recurrent Neural Network / V. Turchenko, L. Grandinetti // 14th IEEE International Conference on Intelligent Engineering Systems. – Las Palmas of Gran Canaria (Spain). – 2010. – P. 25–30.
8. Turchenko V. Efficient Parallelization of Batch Pattern Training Algorithm on Many-core and Cluster Architectures / V. Turchenko, G. Bosilca, A. Bouteiller, J. Dongarra. // 7th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems. – Berlin (Germany). – 2013. – P. 692–698.
9. Turchenko V. Training Algorithm for Radial Basis Function Neural Network / V. Turchenko, V. Golovko, A. Sachenko // 7th International Conference on Neural Networks and Artificial Intelligence. – Minsk (Belarus). – 2012. – P. 47–51.
10. Funahashi K. On the Approximate Realization of Continuous Mappings by Neural Network / K. Funahashi // Neural Networks. – 1989. – No 2. – P. 183–192.
11. Hornik K. Multilayer Feedforward Networks are Universal Approximators / K. Hornik, M. Stinchcombe, H. White // Neural Networks. – 1989. – No 2. – P. 359–366.
12. The MNIST Database of Handwritten Digits [Electronic resource] – Access mode: <http://yann.lecun.com/exdb/mnist/>
13. Hinton G. E. A Fast Learning Algorithm for Deep Belief Nets / G. E. Hinton, S. Osindero, Y. The // Neural Computation. – 2006. – No 18. – P. 1527–1554.
14. Golovko V. Neural Networks: Training, Models and Applications / V. Golovko, A. Galushkin // Moscow: Radiotekhnika, 2001, (in Russian).
15. Turchenko V. Improvement of Parallelization Efficiency of Batch Pattern BP Training Algorithm Using Open MPI / V. Turchenko, L. Grandinetti, G. Bosilca, J. Dongarra // Procedia Computer Science. – 2010. – No 1, Issue 1. – P. 525–533.
16. Open MPI: Open Source High Performance Computing [Electronic resource] – Access mode: <http://www.open-mpi.org/>
17. Turchenko V. Scalability of Parallel Batch Pattern Neural Network Training Algorithm / V. Turchenko // Artificial Intelligence, Journal of Institute of Artificial Intelligence of National Academy of Sciences of Ukraine. – Donetsk. – 2009. – No 2. – P. 144–150.

Рецензія/Peer review : 8.12.2014 p.

Надрукована/Printed : 1.1.2015 p.