УДК 621.391

ДОРОГИЙ Я.Ю.,
ДОРОГА-ІВАНЮК О.О.,
ФЕРЕНС Д.А.

РЕАЛІЗАЦІЯ АЛГОРИТМУ СТРУКТУРНОЇ ОПТИМІЗАЦІЇ НЕЙРОННОЇ МЕРЕЖІ

В статті розглянуте питання реалізації алгоритму структурного оптимізації нейронної мережі та її застосування для задач розпізнавання даних.

In the article the question of implementation of algorithm for neural network architecture optimization and its application for recognition tasks were considered.

Введення в проблему

Апарат нейронних мереж широко використовується для розв'язання різноманітних задач, в тому числі і для задач розпізнавання. Наявність методу автоматичного пошуку оптимальної структури нейронної мережі дала б можливість досліднику швидше отримати структуру нейронної мережі, яка б найкраще відповідали предметній області та наявним вхідним даним.

Аналіз існуючих рішень

Для побудови оптимальної структури нейронної мережі використовується досить широке коло алгоритмів. Першим з таких алгоритмів є алгоритм черепичної побудови [1]. Ідея алгоритму полягає в додаванні нових шарів нейронів таким чином, щоб вхідні навчальні вектори, які мають різні відповідні вихідні значення, мали б в ньому різне внутрішнє представлення. Ще одним яскравим представником є алгоритм швидкої надбудови [2]. Нові нейрони за цим алгоритмом додаються між вихідними шарами. Роль цих нейронів – корегування помилки вихідних нейронів. В загальному вигляді нейронна мережа, що побудована за таким алгоритмом, має форму бінарного дерева. Широко відомими є алгоритми Monoplan, NetLines та NetSphere [3], метод редукції [4]. Але всі ці алгоритми мають достатньо широкий перелік недоліків, тому було вирішено розробити свій алгоритм. В роботі [6] запропоновано алгоритм, реалізацію якого для задач розпізнавання і буде розглянуто в цій роботі.

Мета роботи

Метою даної роботи є аналіз алгоритму структурної оптимізації нейронної мережі в ході її навчання для задач розпізнавання образів та імплементація даного алгоритму програмними засобами.

Алгоритм структурної оптимізації при навчанні

Алгоритм структурного навчання використовується на багатошарових мережах прямого поширення та має ітеративний характер: на кожній ітерації виконується пошук структури мережі кращої за попередню. Пошук мережі виконується шляхом перебору усіх можливих мутацій мережі, вибору та комбінації кращих (селекція та схрещення).

Розглянемо основні параметри алгоритму.

Параметри навчання:

- швидкість навчання, η;
- коефіцієнт інерції, μ;
- коефіцієнт затухання ваг, ε;
- вірогідність активації нейрону прихованого шару, p_h;
- вірогідність активації нейрону вхідного шару, p_i.

Параметри структурного навчання:

- початкова кількість нейронів прихованого шару;
- функція активації прихованого шару;
- функція активації та функція ціни вихідного шару;
- максимальна кількість мутацій при схрещенні;
- кількість епох навчання початкової мережі;
- кількість епох навчання у ітерації;
- види допустимих мутацій;
- частина навчальної вибірки, що використовується для навчання.

Елементарні структурні операції над нейронною мережею

Згідно з [5] введено наступні елементарні структурні операції над мережею:

- додавання синапсу між двома випадково вибраними незв'язаними вузлами або нейронами мережі – операція Syn_{ADD};
- видалення синапсу між двома випадково вибраними незв'язаними вузлами або нейронами мережі – операція Syn_{DEL};
- переміщення синапсу між двома випадково вибраними незв'язаними вузлами або нейронами мережі – операція Syn_{MOD};
- зміна функції активації нейрона для випадково вибраного нейрона – операція A_{MOD};
- серіалізація вузла або нейрона – операції - Ser_{NODE} і Ser_{NR};
- паралелізація вузла або нейрона – операції Par_{NODE} і Par_{NR};
- додавання вузла або нейрона – операції Add_{NODE} і Add_{NR};
- створення нового шару – операція L_{ADD};
- видалення шару НМ – операція L_{DEL}.

Використання чи невикористання наведених структурних операцій залежить від складності поставленої задачі.

Для задач розпізнавання, що будуть наведені в цій статті, використані операції (мутації), описані в [7].

Імплементація алгоритму

Внутрішньо нейронні мережі представлені у вигляді послідовності числових матриць ваг кожного шару окрім вхідного. На рис. 1 зображена послідовність матриць для мережі типу [2-3-2]: матриця 2×3 для прихованого шару та 3×2 для вихідного.

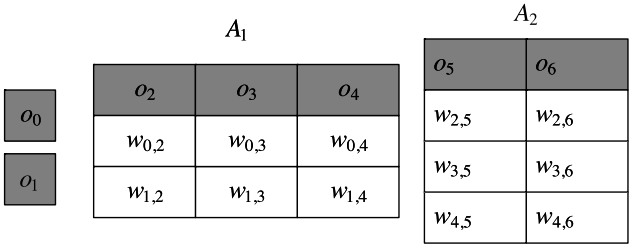

Рис.1. Приклад внутрішньої реалізації для мережі [2-3-2]

Кожен елемент a_{ij} матриці A_k дорівнює значенню ваги між i-тим та j-тим нейронами мережі.

Для реалізації різних видів мутацій використовуються операції над матрицями. При додаванні нових нейронів до шару виконується комбінація із операцій додавання нового рядка та стовпця матриці. На рис. 2, 3 та 4 зображена реалізація додавання нейрону до вхідного, прихованого та вихідного шарів.

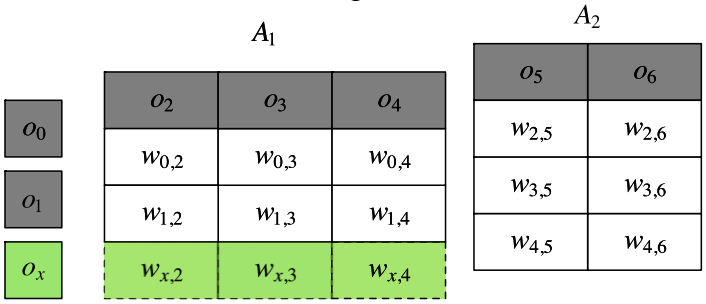

Рис.2. Додавання нейрону до вхідного шару

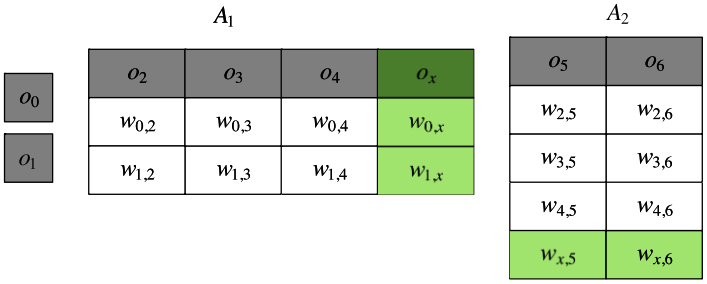

Рис.3. Додавання нейрону до прихованого шару

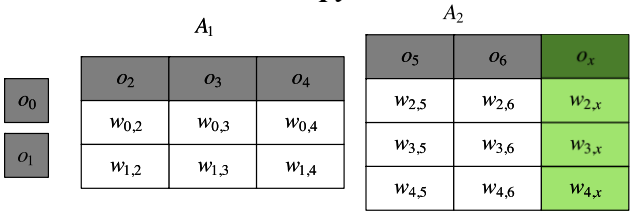

Рис.4. Додавання нейрону до вихідного шару

При видаленні нейронів використовуються протилежні операції. На рис. 5 зображена реалізація видалення 2-го нейрону прихованого шару.

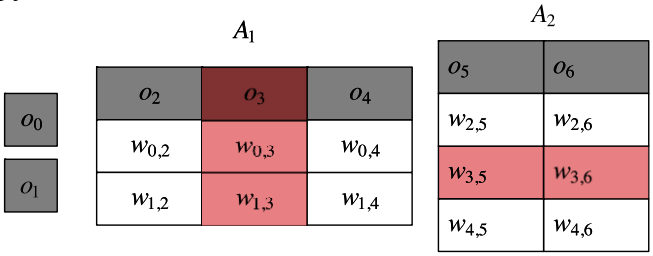

Рис.5. Видалення нейрону прихованого шару

При додаванні нового шару виконується операція вставки нової матриці ваг у послідовність матриць.

Оскільки певні операції змінюють структури матриць, існує певна складність при їх комбінації. Наприклад, при видаленні нейрону o_3 прихованого шару у мережі [2-3-2] у результуючій мережі нейрон o_4 зміститься на 1 позицію

та стане нейроном o_3; при додаванні нового прихованого шару, що містить 4 нейрони перед існуючим прихованим шаром, наступний шар зміститься на 1 позицію. При комбінації різних мутацій їх покрокове виконання необхідно здійснювати у чіткому порядку, що залежить від виду та параметрів кожної мутації. У лістингу 1 нижче наведений фрагмент коду, який правильно виконує комбіновану мутацію. Першим чином виконуються мутації, що не змінюють структури – додавання та видалення зв'язків, згодом виконуються додавання нових нейронів та видалення існуючих; наприкінці виконуються додавання шарів. Мутації, що виконують видалення нейронів, виконуються у порядку зменшення номеру нейрону, аналогічно додавання шарів – у порядку зменшення індексу нового шару.

```
(defmethod mutate ::combined
  [net {:keys [mutations]}]
  (let [grouped-ms (group-by :operation mutations)
        {add-node-ms ::add-node del-node-ms ::del-node
         layer-ms ::add-layer} grouped-ms
        safe-ms (mapcat grouped-ms [::identity ::add-edge ::del-
edge])
        safe-del-node-ms (reverse
                          (sort-by #(second (:deleted-node %)) del-
node-ms))
        safe-layer-ms (reverse
                       (sort-by :layer-pos layer-ms))
        ms (concat safe-ms add-node-ms safe-del-node-ms safe-
layer-ms)]
    (reduce mutate net ms)))
```

Лістинг 1. Фрагмент коду що здійснює комбіновану мутацію

Однією із переваг Clojure над іншими мовами програмування є використання незмінних (immutable) структур даних – колекцій та контейнерів, вміст яких неможливо змінити. Натомість, при спробі додати новий елемент до колекції буде створений новий об'єкт колекції, що містить цей елемент. Операція створення нової колекції оптимізована: обидва об'єкти будуть використовувати частину колекції, що є спільною. На рис. 6 зображений результат додавання об'єкту 5 у кінець масиву [1, 2, 3, 4].

Програмування з використанням незмінних структур даних значно спрощує розуміння програм, а також має наступні переваги:

- простота паралелізації програми – незмінні дані можуть використовуватись паралельно без необхідності синхронізації потоків;
- немає проблем з витоками пам'яті (memory leak);
- простота кешування;
- значна економія пам'яті в деяких випадках.

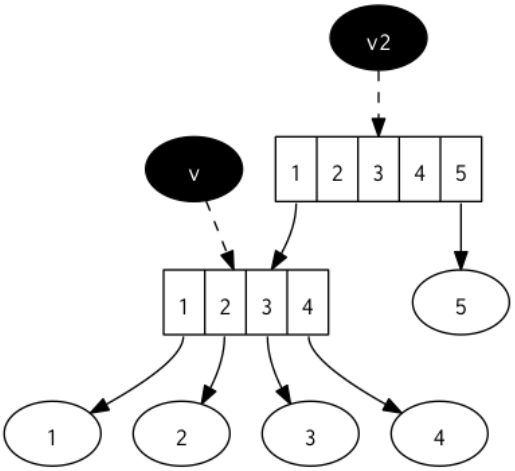

Рис.6. Принцип роботи структур даних у Clojure

Саме завдяки цим властивостям незмінних структур значна частина роботи алгоритму виконується паралельно, максимально використовуючи обчислювальні ресурси.

Розроблена система має архітектуру типу "клієнт-сервер". Діаграма розгортання системи наведена на рис. 7. Загалом, система складається з двох частин:

1. серверний додаток, що виконує навчання нейронної мережі та реалізує алгоритм структурної оптимізації;
2. клієнтський додаток, що реалізує графічний інтерфейс користувача.

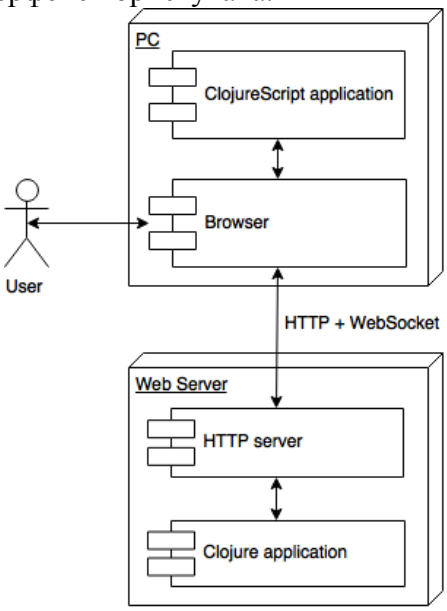

Рис.7. Діаграма розгортання системи

Для реалізації серверного додатку була використана мова програмування Clojure – функціональна мова загального призначення. В яко-

сті середовища виконання була вибрана платформа Java.

Для реалізації графічного інтерфейсу була використана мова програмування ClojureScript – діалект Clojure, що використовується для виконання у середовищі JavaScript.

Експериментальні дослідження

Реалізована програмна система використана для дослідження задачі розпізнавання облич людини. В якості вихідних даних була взята база зображень облич Єльського університету [**Ошибка! Источник ссылки не найден.**].

Формування вибірки. Було вибрано 10 різних осіб та відібрано 50 випадкових зображень кожної особи. Кожне зображення було масштабовано до розміру 26 на 26 пікселів та закодовано у вигляді 676-мірного вектору, значення яскравості пікселів були нормалізовані до діапазону 0..1. Номер кожної особи був закодований вектором розміру 10, що містить 9 нулів та одиницю в позиції номеру персони. Отримані 500 прикладів були випадково поділені на навчальну та тестувальну вибірки у відношенні 2:1.

На рис. 8 зображені вихідні зображення та зображення, що використовуються для навчання нейронної мережі.

Рис.8. Приклад формування набору даних

Архітектура початкової мережі. Для оцінки роботи алгоритму була використана мережа, архітектура якої зображена на рис. 9.

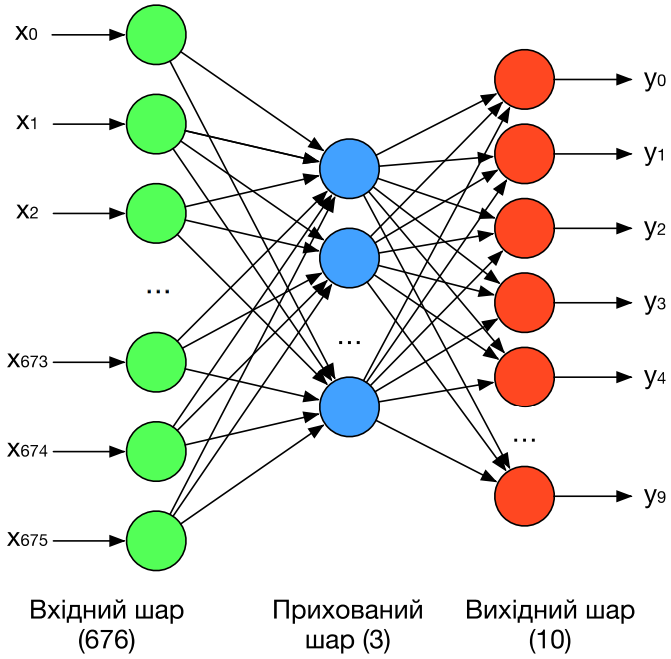

Рис.9. Архітектура мережі для розпізнавання зображень

Дослідження роботи алгоритму. Були використані наступні значення параметрів навчання:

- швидкість навчання: $\eta = 0.002$;
- коефіцієнт інерції: $\mu = 0.1$;
- коефіцієнт затухання ваг: $\varepsilon = 0.1$;
- вірогідність активації нейрону прихованого шару: $p_h = 1$;
- вірогідність активації нейрону вхідного шару: $p_i = 1$.

Параметри алгоритму вибрані наступні:

- початкова кількість нейронів прихованого шару: 3;
- функція активації прихованого шару: ReLU;
- функція активації та функція ціни вихідного шару: softmax;
- максимальна кількість мутацій при схрещенні: $M = 50$;
- кількість епох навчання початкової мережі: $T_0 = 100$;
- кількість епох навчання у ітерації: $T_i = 5$;
- види допустимих мутацій: додавання та видалення синапсів;
- частина навчальної вибірки що використовується для навчання: 1.

Протягом 40 ітерацій алгоритму було здійснено 300 видалень та 128 додавань синапсів. На рис. 10 та 11 зображені залежності значення

ціни та точності класифікації від кількості виконаних епох навчання, отримана точність розпізнавання наведена в табл.1.

Табл.1. Результуюча точність класифікації зображень для $T_i = 5$

Тип НМ	Навчання, %	Тестування, %
Звичайна	97.59	93.41
Оптимізована	98.19	95.80

Завдяки оптимізації структури зв'язків вдалось знизити відсоток помилкової класифікації з 6.6% до 4.2% на тестувальній вибірці.

Також, був проведений експеримент, при якому прийнято $T_i = 3$, зображено на рис. 12 та 13. Протягом 100 ітерацій алгоритму було здійснено 645 видалень та 457 додавань синапсів. Відсоток помилкового розпізнавання вдалось знизити з 7.8% до 6.0%. на тестувальній вибірці. Результати класифікації наведені в табл.2.

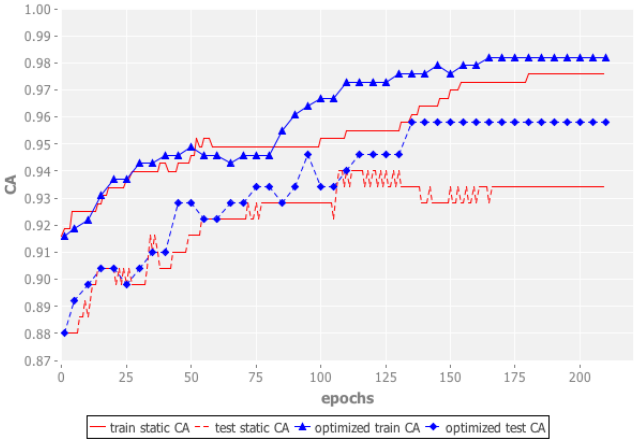

Рис.10. Точність класифікації зображень для $T_i = 5$

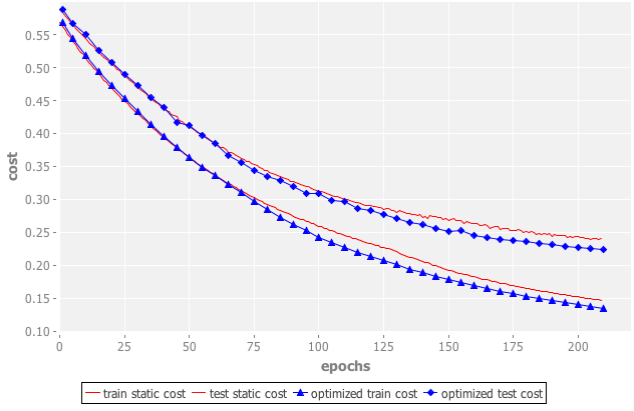

Рис.11.Значення ціни при для класифікації зображень для $T_i = 5$

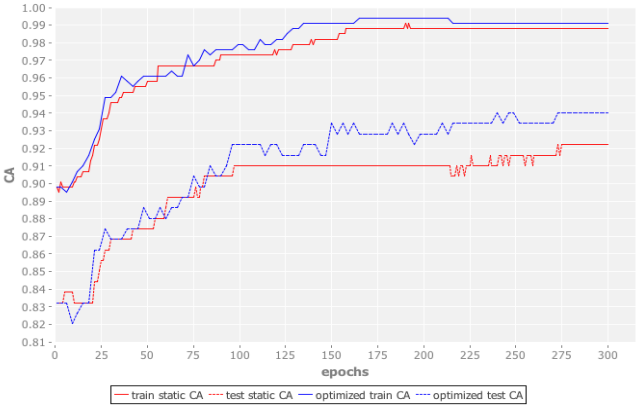

Рис.12. Точність класифікації зображень для $T_i = 3$

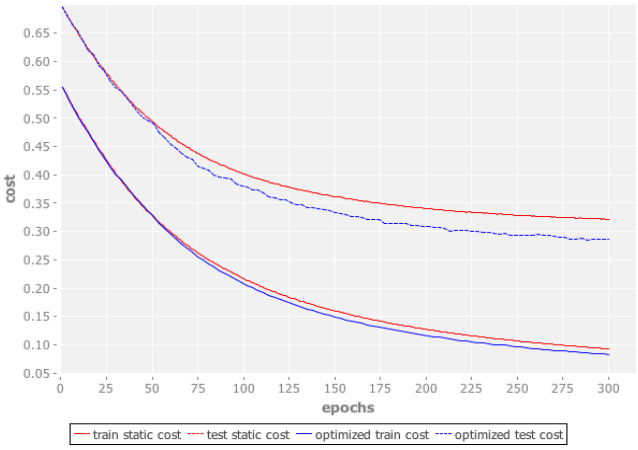

Рис.13. Значення ціни при для класифікації зображень для $T_i = 3$

Табл.2. Результуюча точність класифікації зображень для $T_i = 3$

Тип НМ	Навчання, %	Тестування, %
Звичайна	98.79	92.21
Оптимізована	99.09	94.01

Висновки

В роботі розглянуті питання імплементації алгоритму структурної оптимізації, який запропонований в [6], а також проаналізована можливість використання даного алгоритму для задач розпізнавання образів..

Список посилань

1. Mezard M., and Nadal J.P. Learning in feedforward layered networks: The Tiling algorithm // Journal of Physics. – 1989. – V. A22. – P. 2191 - 2203,
2. Frean M. The Upstart Algorithm: A Method for Constructing and Training Feed-Forward Neural Networks // Tech. Rep. 89/469, Edinburgh Univ., 1989.

3. Ash T. Dynamic Node Creation in Back-Propagation Networks // Connection Science. – 1989. – V. 1.
4. Mozer M.C., Smolensky P. Skeletonization: a technique for trimming the fat from a network via relevance assessment // Advances in Neural Information Processing Systems. – 1989. – V. 1. – P. 107 - 115.
5. Дорогий Я.Ю. Ускоренный алгоритм обучения сверточных нейронных сетей / Я.Ю. Дорогий // Вісник НТУУ «КПІ», «Інформатика, управління та обчислювальна техніка», №57. – 2012. – С. 150-154.
6. Дорогий Я.Ю. Алгоритм структурної оптимізації нейронної мережі / Я.Ю.Дорогий // Вісник НТУУ «КПІ», «Інформатика, управління та обчислювальна техніка», №61. – 2014. – С. 169-173.
7. Дорогий Я.Ю. Застосування алгоритму структурної оптимізації нейронної мережі в задачах класифікації даних / Я.Ю.Дорогий, В.В.Цуркан, О.О.Дорога-Іванюк, Д.А.Ференс// Вісник НТУУ «КПІ», «Інформатика, управління та обчислювальна техніка», №62. – 2015. – С. 169-173.