УДК 621.391

ДОРОГИЙ Я.Ю.

АЛГОРИТМ СТРУКТУРНОЇ ОПТИМІЗАЦІЇ НЕЙРОННОЇ МЕРЕЖІ

В статті розглянуте питання архітектурної оптимізації нейронної мережі. Запропонований алгоритм структурної оптимізації в ході навчання нейронної мережі.

In the article the question of neural network architecture optimization was considered. The algorithm of structure optimization on learning stage was offered

Введення в проблему

На сьогоднішній день апарат нейронних мереж широко використовується для розв'язання різноманітних задач. В зв'язку з цим, питання розробки алгоритму навчання, що мав би смогу динамічно оптимізувати структуру нейронної мережі, є дуже актуальним. Наявність такого методу 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 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 досить широкий перелік недоліків, тому було вирішено розробити свій алгоритм.

Мета роботи

Метою даної роботи є створення алгоритму структурної оптимізації нейронної мережі в ході її навчання.

Графова модель представлення нейронної мережі

Будь-яку нейронну мережу можна представити у вигляді графу. Багатошарова архітектура нейронної мережі представляє собою ациклічний граф G, представлений як набір вузлів L_i^j, згрупованих у кластери за шарами та зв'язків між ними. Приклад такого представлення нейронної мережі показано на рис. 1.

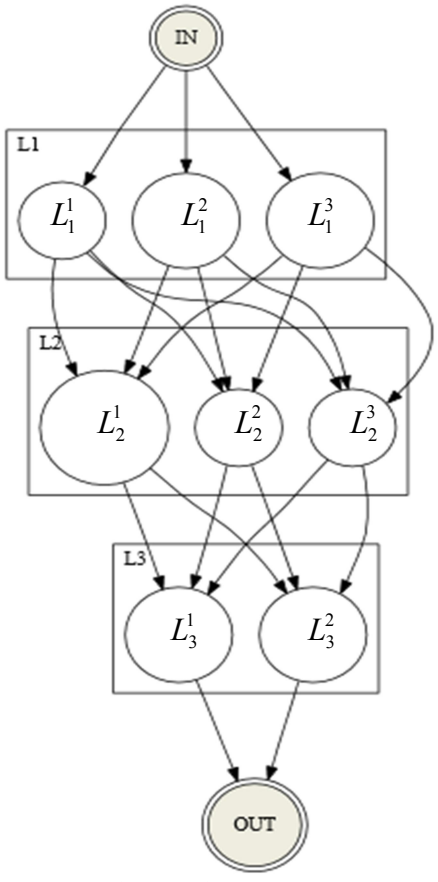

Рис.1. Графова модель НМ

Кожний вузол L_i^j представляє собою або елементарний нейрон, або елемент більш складної архітектури, наприклад, невелику нейронну мережу (рис. 2).

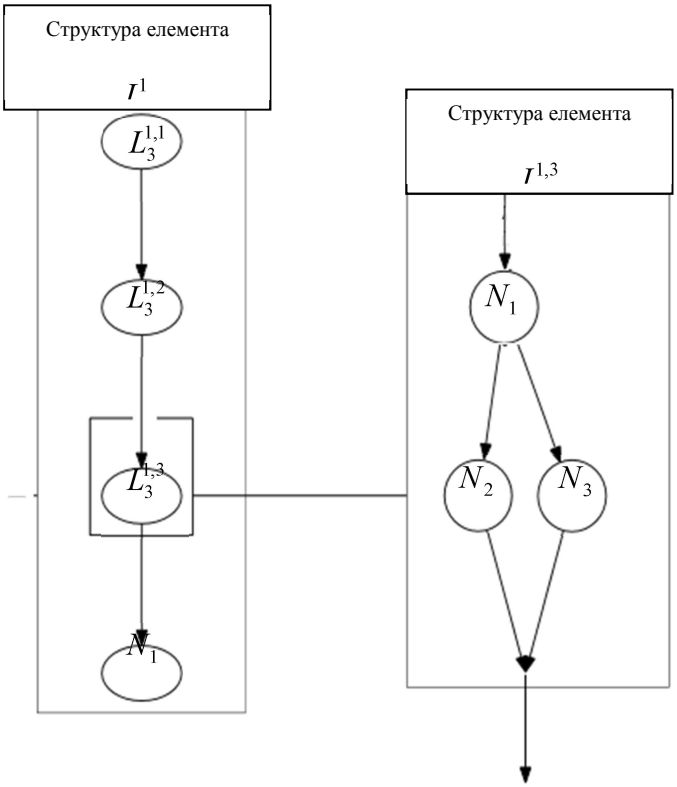

Рис.2. Структура вузла

Елементарні структурні операції над нейронною мережею

Введемо 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}.

Розглянемо операції більш детально.

Нехай нашу мережу задано наступним графом:

G = {V, E}, де

V – множина вузлів та нейронів;

E – множина синапсів, що їх зв'язуються;

V_A – підмножина вузлів, що належить попередньому підрівню (тобто такі, з яких виходять синапси);

V_B – підмножина вузлів, що належить наступному підрівню (тобто такі, в які входять синапси).

Операція додавання синапсу між двома випадково вибраними незв'язаними вузлами або нейронами мережі проілюстрована на рис. 3.

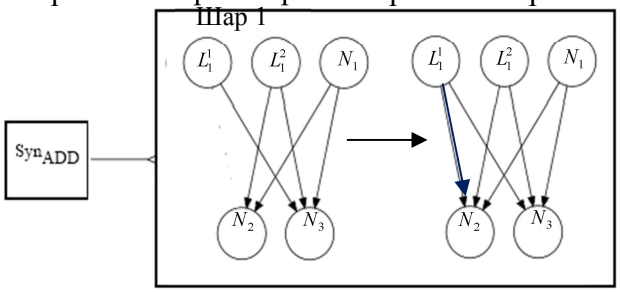

Рис.3. Операція Syn_{ADD}

Тобто,

$E(G) = E(G) \cup (l, n)$

$V_A = \{L_1^1, L_1^2, L_1^3\}, V_B = \{N_1, N_2\},$

$l \in V_A, n \in V_B.$

Операція видалення синапсу між двома випадково вибраними зв'язаними вузлами або нейронами мережі представлена на рис. 4.

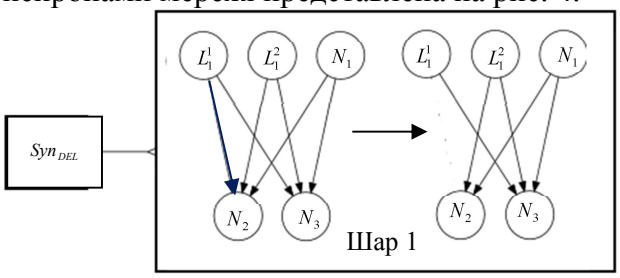

Рис.4. Операція Syn_{DEL}

Тобто,

$E(G) = E(G) \setminus (l, n)$

$V_A = \{L_1^1, L_1^2, L_1^3\}, V_B = \{N_1, N_2\},.$

$l \in V_A, n \in V_B.$

Операція переміщення синапсу між двома випадково вибраними вузлами або нейронами мережі представлена на рис. 5.

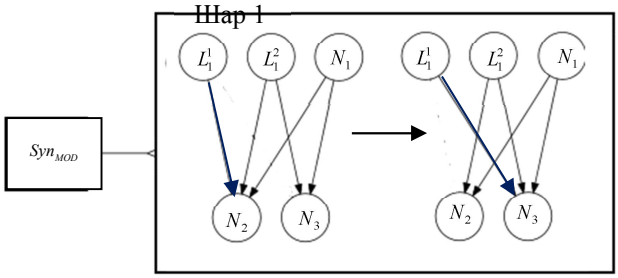

Рис.5. Операція Syn_{MOD}

Тобто,

$E(G) = E(G) \setminus (l, n);$

$E(G) = E(G) \cup (l, m);$

$V_A = \{L_1^1, L_1^2, L_1^3\}, V_B = \{N_1, N_2\},$

$l \in V_A, n, m \in V_B.$

Операція серіалізації Ser_{NODE} і Ser_{NR} випадково вибраного вузла або нейрона мережі представлена на рис. 6.

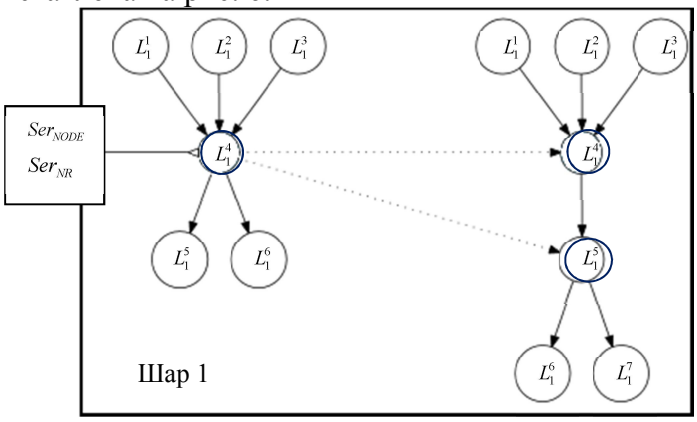

Рис.6. Операція серіалізації

Тобто,

$E(G) = E(G) \setminus \forall (l, n);$

$V_A = \{L_1^4\}, V_B = \{L_1^5, L_1^6\}, l \in V_A, n \in V_B.$

$V(G) = V(G) \cup L_1^5$ (переіндексація)

$V_B' = \{L_1^6, L_1^7\}, V_A' = \{L_1^5\}$

$E(G) = E(G) \cup (l, m); l \in V_A', m \in V_B';$

$E(G) = E(G) \cup (k, p); k \in V_A, p \in V_A'.$

Операція паралелізація Par_{NODE} і Par_{NR} випадково вибраного вузла або нейрона мережі представлена на рис. 7.

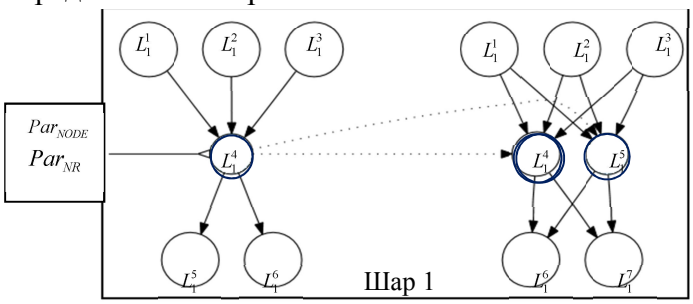

Рис.7. Операція паралелізації

Тобто,

$V(G) = V(G) \cup L_1^5$ (переіндексація)

$V_B' = \{L_1^6, L_1^7\}, V_A = \{L_1^1, L_1^2, L_1^3\}, V_A' = \{L_1^5\}$

$E(G) = E(G) \cup (l, m); l \in V_A, m \in V_A';$

$E(G) = E(G) \cup (k, p); k \in V_A', p \in V_B'.$

Операція додавання вузла або нейрона Add_{NODE} і Add_{NR} проілюстрована на рис. 8.

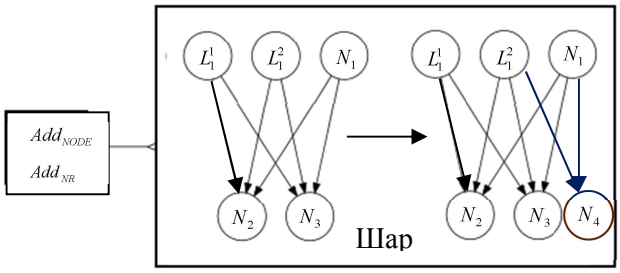

Рис.8. Операція Add_{NODE} ***і*** Add_{NR}

При додаванні нового вузла чи нейрона випадковим чином додаються декілька синапсів від створеного нейрона та до нього.

При видаленні вузла чи нейрона видаляються самі нейрони та вузли, а також всі синапси, що входять в даний елемент. Всі зв'язки, що виходять з даного елементу випадковим чином з'єднуються з елементами, що йдуть за видаленим.

Операція створення нового шару L_{ADD} є модифікатором для розглянутих вище операцій додавання вузла або нейрона та операцій серіалізації і паралелізації і вказує на те, що результат операції додається в новий шар мережі.

Алгоритм структурної оптимізації при навчанні

Розглянемо алгоритм структурної оптимізації при навчанні. В якості базового алгоритму навчання візьмемо алгоритм зворотного поширення помилки, запропонований в [5].

НМ, що розглядається, має сигмоїдальну функцію активації:

$$f(u) = \frac{1}{1 + e^{-u}}, \quad (1)$$

де

$$u = w_0 + w_1 x_1 + ... + w_n x_n. \quad (2)$$

Введемо наступні позначення: множина навчальних даних $X_s = \{x_i \mid i = 1,...,N\}$ та відповідні їм очікувані значення $Y_s = \{y_i \mid i = 1,...,M\}$, де N – число входів НМ, M – число виходів НМ, $s = 1,..,S$, а S – вимірність навчальної вибірки. Позначимо множину значень виходів нейронів

НМ через $F_i = \{f_j^i\}$, де і – номер шару, а j – номер нейрона в шарі. Для вхідного шару $j = 1,2,...,N$, для першого прихованого шару – $j = 1,2,...,C_1$, другого – $j = 1,2,...,C_2$ і т.д., для останнього прихованого шару – $j = 1,2,...,M$. Загальна кількість шарів НМ дорівнює К та кількість нейронів в усіх прихованих шарах буде рівна $C = \{C_0, C_1, ...C_K\}$, де $C_0 = N, C_K = M$.

Алгоритм зворотнього поширення помилки можна представити наступним чином:

1) Для вхідного шару встановлюємо значення кожного елементу відповідно з вхідним вектором. Значення виходу кожного елемента встановимо рівним входу.

$$F_0 = X_1. \qquad (3)$$

2) Для нейронів першого шару обчислюємо загальний вхід та вихід:

$$u_j^1 = w_{j,0}^1 + \sum_{i=1}^{N} f_i^0 w_{j,i}^1, \qquad (4)$$

де

$$f_j^1 = \frac{1}{1 + e^{-u_j^1}}, \qquad (5)$$

де $j = 1,2,..., N_1$ – номер нейрона першого прихованого шару, $w_{j,i}^l$ – і-й ваговий коефіцієнт j-го нейрона l–шару НМ, f_i^0 – значення і-го нейрона вхідного шару.

3) Повторюємо перший крок для всіх прихованих шарів НМ, включаючи вихідний шар нейронів, причому формули (4) та (5) трохи зміняться, та будуть мати вигляд:

$$u_j^k = w_{j,0}^k + \sum_{i=1}^{C_k} f_i^{c-1} w_{j,i}^c, \qquad (6)$$

де

$$f_j^c = \frac{1}{1 + e^{-u_j^c}}, \qquad (7)$$

де $c = 2,..., K$.

4) Порівнюємо значення векторів Y_0 та F_K, якщо різниця між вектором – зразком Y_0 та реальним виходом НМ F_K знаходиться в допустимому діапазоні, то переходимо до кроку 5, якщо ні, то переходимо до кроку 6.

5) Для вхідного вектора встановлюємо значення кожного елемента, що дорівнює відповідному елементу $X_2 (F_0 = X_2)$ та знову переходимо до кроку 2. Якщо ж на вхід НМ подано останній вектор X_S, то в залежності від того, чи є необхідним навчання на цій ітерації, або знову подаються на вхід усі навчальні вектори, починаючи з першого, або вважається, що НМ навчена і навчання закінчується.

6) Для кожного нейрона вихідного шару обчислюється помилка. Оскільки розглядається НМ з сигмоїдальною функцією активації, то значення помилки дорівнює

$$\delta_i = \begin{cases} f_i^{'1}(1 - f_i^{'1})(y_i - f_i^{'1}) \; при \; \mathrm{f_i} < f_i^{'1} \\ \mathrm{f_i}(1 - \mathrm{f_i})(y_i - \mathrm{f_i}) \; при \; f_i^{'2} < \mathrm{f_i} < f_i^{'1} \\ f_i^{'2}(1 - f_i^{'2})(y_i - f_i^{'2}) \; при \; \mathrm{f_i} > f_i^{'2} \end{cases}, \qquad (8)$$

де f_i – отримане значення j-го елемента вихідного шару; $f_i^{'1}, f_i^{'2}$ – точки локальних екстремумів для очікуваного значення y_i в діапазоні [0,1].

7) Розраховується коефіцієнт мутації мережі:

$$M_{NET} = \frac{1}{\delta(t+1)}, \qquad (9)$$

В залежності від розрахованого коефіцієнта виконується модифікація структури нейронної мережі за допомогою використання елементарних структурних операцій, розглянутих раніше (див. табл. 1). Тренд розрахованого коефіцієнта вказую на динаміку навчання. Якщо з кожною епохою навчання відбувається його зростання, тобто помилка зменшується, не має сенсу щось міняти в структурі мережі. І, навпаки, якщо тренд падає, потрібно модифікувати структуру мережі.

Табл. 1. Використання елементарних структурних операцій

Значення M_{NET}	Тренд M_{NET}	Операція
>1	Зростає	-
>1	Падає	L_{ADD}
>100	Зростає	-
>100	Падає	Ser_{NODE}, Ser_{NR}, Par_{NODE}, Par_{NR}, Add_{NODE}, Add_{NR}
>500	Зростає	-
>500	Падає	Syn_{ADD}, Syn_{MOD}, A_{MOD}

7) Для передостаннього шару обчислюється помилка кожного нейрона

$$\delta_i^{K-1} = f_i^{K-1}(1 - f_i^{K-1}) \sum_{j=1}^{C_K} \delta_j^K w_{i,j}^{K-1}, \qquad (10)$$

де δ_j^K – значення помилки j-го елементу К-го шару, $w_{i,j}^{K-1}$ – вага зв'язку між і-м елементом К-1 шару та j-м елементом К-го шару, f_i^{K-1} – значення і-го елемента К-1-го шару.

8) Для всіх інших прихованих шарів помилка обчислюється за формулою (10).

9) Далі, для всіх шарів поновлюються значення вагових коефіцієнтів кожного нейрона

$$\Delta w_{i,j}^c(t+1) = \eta(\delta_i^c f_j^{c-1}) + \alpha \Delta w_{i,j}^c(t), \qquad (11)$$

де η – швидкість навчання, α – інерція, або вплив значення попередньої зміни, зазвичай береться значення $\alpha < 1$, t – номер ітерації. І тоді встановлюємо нові значення вагових коефіцієнтів, рівними

$$w_{i,j}^c = w_{i,j}^c + \Delta w_{i,j}^c(t) \qquad (12)$$

10) Для вхідного вектора встановлюємо значення кожного нейрона, що дорівнює відповідному елементу X_2 ($F_0 = X_2$) та знову виконуємо перехід до кроку 2. Якщо вже поданий останній вектор X_S на вхід НМ, то знову подається перший навчальний вектор.

Висновки

В ході дослідження побудований алгоритм структурної оптимізації нейронної мережі. Також, проведений ряд дослідів побудованого алгоритму, які доводять можливість використання його в задачах розпізнавання та класифікації даних.

Список використаної літератури

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.