ИНСТИТУТ ПО БИОФИЗИКА И БИОМЕДИЦИНСКО ИНЖЕНЕРСТВО
БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ

МОДЕЛИРАНЕ НА НЕВРОННИ МРЕЖИ ЧРЕЗ ОБОБЩЕНИ МРЕЖИ

АВТОРЕФЕРАТ

на дисертационен труд за присъждане на образователна и научна степен „доктор”
професионално направление: 4.6 „Информатика и компютърни науки“, научна специалност: 02.21.10 „Приложение на принципите и методите на кибернетиката“

инж. маг. Антон Жоров Антонов

Научни ръководители:

чл.-кор. проф. дмн дтн Красимир Атанасов
проф. дтн Стефан Хаджитодоров

София
2014

Защитата на дисертационния труд ще се състои на ..... от ..... в заседателната зала на Институт по биофизика и биомедицинско инженерство – Българска академия на науките (ул. "Акад. Г. Бончев бл. 105").

Дисертационният труд е обсъждан на разширен научен семинар на секция "Биоинформатика и математическо моделиране" на ИБФБМИ на 11.12.2014 г.

Дисертационният труд съдържа увод, 5 глави и научно-приложни приноси и е в обем от 187 страници. Цитирани са 161 литературни източника.

Автор: Антон Жоров Антонов

ЗАГЛАВИЕ: Моделиране на невронни мрежи чрез обобщени мрежи

Характеристика на дисертационния труд

Актуалност на проблема

В продължение на векове алгоритмите в математиката биват последователни. През 1962 г. се появяват мрежите на Петри, които са първият математически апарат, позволяващ описването на конкурентни процеси в математиката, а десет години по-късно получават своите първи неконсервативни разширения - E-мрежи [117], мрежи на Петри с време [126], цветни мрежи на Петри [71], стохастични мрежи [115, 130] и други. Още десет години по-късно са дефинирани обобщените мрежи [15], които от своя страна дават възможност да бъдат симулирани конкурентни процеси и събитийно управлявани симулации.

От няколко десетилетия се развиват компютърни системи, позволяващи паралелна обработка на данните, а едва в последното десетилетие истинската паралелна обработка на процесорите навлиза в ежедневието [158]. Имайки предвид тези факти, може да се приеме, че пред учените от различни области заедно с потребителите на съвременните компютърни системи предстои търсенето на нови начини за максимално ефективно използване на появилите се възможности за паралелно и конкурентно пресмятане на всеки един процесор, във всяко едно устройство – от мобилните устройства с многоядрени процесори до облачните системи.

Алгоритмите за машинно обучение заедно с невронните мрежи започват своето развитие преди повече от 70 години [108], представяйки в опростен вариант работата на човешкия неврон. Въпреки многото разработки на невронни мрежи и техните модификации, които според IEEE Xplorer Digital Library (http://ieeexplore.ieee.org/Xplore/home.jsp) са повече от 100 000 статии и 7000 от тях са свързани с паралелни или конкурентни реализации на невронни мрежи, реално в много малък процент от практическите разработки се използва паралелна или конкурентна реализация. Имайки предвид тенденцията от последното десетилетие, законът на Мур много трудно, почти невъзможно, ще продължава да е валиден при непрекъснатото увеличаване на честотите на процесорите. Този факт, както и тенденцията за увеличаване на броя на ядрата във всеки един процесор водят до извода, че е необходимо да се търсят нови методи за максимално използване на предоставените паралелни изчислителните ресурси.

Използването на паралелните изчислителни ресурси в невронните мрежи и алгоритмите за машинно обучение със серийни данни е предстоящата вълна на развитие, която трябва да посрещне изискванията за работа с големите по обем данни от разнообразни източници при високи скорости на обработка (Big Data). Появата на обеми от данни от такава величина, които трябва да бъдат обработени (достигащи до 10^{15} байта) [28, 104, 105] изисква създаването на нови подходи и алгоритми за обработка на данните, които са същевременно основен обект на изследване, разработка и симулация в настоящия дисертационен труд.

Цел на дисертационния труд:

Проучване, изследване и оптимизиране на невронни мрежи, използвайки апарата на обобщените мрежи.

Задачи на дисертационния труд:

1. Въз основа на резултатите от направения литературен обзор на публикациите по невронни мрежи да бъдат разработени модели на невронните мрежи, които до момента не са били симулирани с обобщени мрежи.
2. Да бъде разработен модел на неврон, който да бъде използван за основа в останалите модели на невронни мрежи.
3. Оценявайки предимствата на обобщените мрежи при работа с паралелни процеси, да се създадат конкурентни и оптимизационни модели на невронни мрежи с цел максимално използване на широко достъпните паралелни изчислителни ресурси.
4. Да се потърсят нови алгоритми за конкурентно или паралелно обучение на вече съществуващи невронни мрежи с цел ускорение на обучителния процес.

Апробация на резултатите

Дисертационният труд е представен на разширен научен семинар на секция "Биоинформатика и математическо моделиране" на ИБФБМИ и на следните международни конференции, статии, научни списания и годишници: "Issues in the Representation and Processing of Uncertain Imprecise Information: Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets, and Related Topics", "Advanced Studies in Contemporary Mathematics", IEEE 6th International Conference 'Intelligent Systems', 2012, Годишник на секция „Информатика", Съюз на учените в България, "Comptes Rendus de L'Academie Bulgare des Sciences".

Списък на публикациите по дисертационния труд

1. Antonov, A., Presentation of neuron by generalized net, Issues in the Representation and Processing of Uncertain Imprecise Information: Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets, and Related Topics (K. Atanassov, J. Kacprzyk, M.Krawczak and E. Szmidt, Eds.), Akademicka Oficyna Wydawnictwo EXIT, Warszawa, 2005, 1-10.
2. Atanassov, K., S. Sotirov, A. Antonov, Generalized net model for parallel optimization of feed-forward neural network, Advanced Studies in Contemporary Mathematics, Vol.15, No. 1, 2007, 109-119.

3. Antonov, A., S. Hadjitodorov, Concurrent Algorithm for Learning of Neural Networks, IEEE 6th International Conference 'Intelligent Systems', 2012, 225-228.
4. Антонов, А., Обзор на обобщеномрежовите модели, описващи функционирането на невронни мрежи, Годишник на секция „Информатика", Съюз на учените в България, Том 5, 2012, 120-128.
5. Antonov, A., Generalized net model for parallel optimization of hidden units in neural networks with radial basis functions, Comptes Rendus de L'Academie Bulgare des Sciences, Vol. 66, No. 9, 2013, 1239-1246.

Списък на забелязани цитирания на публикации по дисертационния труд

Забелязани са общо 4 цитирания на 3 публикации.

Antonov, A., Presentation of neuron by generalized net, Issues in the Representation and Processing of Uncertain Imprecise Information: Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets, and Related Topics (K. Atanassov, J. Kacprzyk, M.Krawczak and E. Szmidt, Eds.), Akademicka Oficyna Wydawnictwo EXIT, Warszawa, 2005, 1-10.

1. Alexieva, J., E. Choy, E. Koycheva, Review and bibloigraphy on generalized nets theory and applications. - In: A Survey of Generalized Nets (E. Choy, M. Krawczak, A. Shannon and E. Szmidt, Eds.), Raffles KvB Monograph, No 10, 2007, 207-301.
2. Sotirov, S., K. Atanassov, Generalized Nets In Artificial Inteligence Volume 6 : Generalized Nets And Supervised Neural Networks, Prof. Marin Drinov ACADEMIC PUBLISHING HOUSE, Sofia, 2012.

Antonov, A., S. Hadjitodorov, Concurrent Algorithm for Learning of Neural Networks, IEEE 6th International Conference 'Intelligent Systems', 2012, 225-228.

3. Sotirov, S., K. Atanassov, Generalized Nets In Artificial Inteligence Volume 6 : Generalized Nets And Supervised Neural Networks, Prof. Marin Drinov ACADEMIC PUBLISHING HOUSE, Sofia, 2012.

Антонов, А., Обзор на обобщеномрежовите модели, описващи функционирането на невронни мрежи, Годишник на секция „Информатика", Съюз на учените в България, Том 5, 2012, 120-128.

4. Sotirov, S., K. Atanassov, Generalized Nets In Artificial Inteligence Volume 6 : Generalized Nets And Supervised Neural Networks, Prof. Marin Drinov ACADEMIC PUBLISHING HOUSE, Sofia, 2012.

Обем и структура на дисертационния труд

Дисертационният труд е в обем от 187 страници и се състои от увод, пет глави, научни и научно-приложни приноси, декларация за оригиналност

на резултатите, библиография, списък на публикациите по дисертационния труд и списък на забелязаните цитирания на публикации по дисертационния труд. Дисертационният труд включва 57 фигури и 51 таблици, а библиографията към него – 161 заглавия.

КРАТКО СЪДЪРЖАНИЕ НА ДИСЕРТАЦИОННИЯ ТРУД

В настоящия автореферат са използвани номерации на глави, подточки, фигури, таблици и цитирана литература, съответстващи на тези в дисертационния труд.

Глава 1. Въведение в теориите на обобщените и невронните мрежи

1.1 Обобщени мрежи

1.1.1 Дефиниция на понятието обобщена мрежа

1.1.1.2 Формална дефиниция на ОМ

Съгласно [19, 2, 22] всеки преход се задава чрез наредена седморка от вида:

$$Z = <L', L'', t_1, t_2, r, M, \square>,$$

където:

- $L' = \{l'_1, ..., l'_i, ..., l'_m\}$ – крайно непразно множество от входните позиции на прехода;
- $L'' = \{l''_1, ..., l''_j, ..., l''_n\}$ – крайно непразно множество от изходните позиции на прехода;
- t_1 – момент от време на активиране на прехода;
- t_2 – задава продължителността на активното състояние на прехода;
- r – условие на прехода, определящо кои ядра могат да преминат от входните към изходните му позиции. То се задава чрез индексирана матрица от вида:

$$r = \begin{array}{c|cccccc} & l''_1 & \cdots & l''_j & \cdots & l''_n \\ \hline l'_1 & r_{1,1} & \cdots & r_{1,j} & \cdots & r_{1,n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ l'_i & r_{i,1} & \cdots & r_{i,j} & \cdots & r_{i,n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ l'_m & r_{m,1} & \cdots & r_{m,j} & \cdots & r_{m,n} \end{array},$$

където $r_{i,j}$ е предикат, съответстващ на $i^{та}$ входна позиция на прехода и $j^{та}$ изходна позиция на прехода. Ако предикатът е верен (има стойност

"истина"), е възможно преминаване на ядро от $i^{та}$ към $j^{та}$ позиция. Предикатите не могат да зависят от бъдещи събития.

- M – индексирана матрица на капацитетите на дъгите, имаща вида:

$$M = \begin{array}{c|cccccc} & l''_1 & \cdots & l''_j & \cdots & l''_n \\ \hline l'_1 & m_{1,1} & \cdots & m_{1,j} & \cdots & m_{1,n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ l'_i & m_{i,1} & \cdots & m_{i,j} & \cdots & m_{i,n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ l'_m & m_{m,1} & \cdots & m_{m,j} & \cdots & m_{m,n} \end{array},$$

където $m_{i,j}$ е естествено число, задаващо капацитета на дъгата от $i^{та}$ входна позиция на прехода към $j^{та}$ изходна позиция на прехода, а може да приема и стойност „∞" в случаите когато капацитетът на дъгата не се явява условие за функционирането на модела;

- □ – обект, имащ вид, подобен на булев израз. В него участват идентификаторите на всички входни позиции на прехода, свързани с логическите операции "и" ($\wedge$) и "или" ($\vee$). Когато стойността на типа, изчислен като булев израз, е "истина", съответният преход може да се активира, в противен случай – не. Ако булевият израз е от вида: $\wedge(l_{i_1}, l_{i_2},\ldots, l_{i_u})$, то това означава, че във всяка входна позиция $l_{i_1}, l_{i_2},\ldots, l_{i_u}$ трябва да има най-малко по едно ядро. Ако булевият израз е от вида: $\vee(l_{i_1}, l_{i_2},\ldots, l_{i_u})$, то това означава, че най-малко в една от входните позиции $l_{i_1}, l_{i_2},\ldots, l_{i_u}$ трябва да има поне едно ядро.

След като е дефинирано понятието *преход* като основен градивен елемент на обобщената мрежа, може да се дефинира и понятието обобщена мрежа. Наредената четворка:

$E = <<A, \pi_A, \pi_L, c, f, \theta_1, \theta_2>, <K, \pi_K, \theta_K>, <T, t^\circ, t^*>, <X, \Phi, b>>$,

се нарича *обобщена мрежа (ОМ),* ако:

- A – множеството от всички преходи в мрежата;
- π_A – функция, задаваща приоритетите на преходите, т.е. π_A: $A \to N$, където $N = \{0, 1, 2,\ldots\} \cup \{\infty\}$;
- π_L – функция, задаваща приоритетите на позициите, като L е множеството от всички позиции на обобщената мрежа.

  π_L: $L \to N$, където $L = pr_1A \cup pr_2A$ и с pr_iX е означена $i^{-та}$ проекция на n-мерното множество X, $n \in N$, $1 \le i \le n$.

- c – функция, задаваща капацитетите на позициите, т.е. c: $L \to N$;
- f – функция, определяща вярностната стойност на предикатите (приема стойности "лъжа" или "истина", или елементите на множество {0,1}, но за някои разширения на ОМ може да приема стойности в интервала [0,1] или в [0,1]×[0,1], вж. т. 1.1.7);
- θ_1 – функция, задаваща следващия момент от време, в който може да се активира преходът. Стойността на тази функция се преизчислява в момента, в който завършва активното състояние на прехода. $\theta_1(t) = t'$, където $pr_3Z = t$, $t' \in [T, T + t^*]$ и $t' \ge t$, т.е. $t' \in [t, T + t^*]$;

- θ_2 – функция, която задава продължителността на активното състояние на даден преход Z, и стойността ѝ се изчислява в момента, в който се активира преходът; $\theta_2(t) = t'$, където $pr_4 Z = t \in [T, T + t^*]$ и $t' \geq 0$;
- K – множество на ядрата в ОМ;
- π_K – функция, която задава приоритетите на ядрата т.е. π_K: $K \rightarrow N$;
- θ_K – функция, която задава момента от време, в който определено ядро може да влезе в обобщената мрежа, т.е. $\theta_K(\alpha) = t$, където $\alpha \in K$, $t \in [T, T+t^*]$;
- T – момент от време, в който обобщената мрежа започва да функционира. Моментът T се определя по фиксирана времева скала;
- t° – елементарната времева стъпка на фиксираната времева скала;
- t^* - продължителност на функционирането на обобщената мрежа;
- X – множество на началните характеристики, с които ядрата влизат в мрежата;
- Φ – характеристична функция. Тя определя новата характеристика на ядрото при преместването му от входната позиция на даден преход в изходната;
- b: $K \rightarrow N$ – функция, задаваща максималния брой характеристики, които едно ядро може да получи по време на движението си в обобщената мрежа. Ако за ядро σ е изпълнено $b(\sigma)=1$, то това ядро ще влезе в мрежата с *начална (нулева) характеристика.* След това то ще помни само нея и *последната (текущата) си* характеристика. Ако $b(\sigma)=k$ $(k<\infty)$, то ядрото σ помни последните си k характеристики, както и началната си характеристика (която винаги се пази). Ако $b(\sigma)=\infty$, то ядрото σ помни всичките си получени характеристики.

В описанието на дадена ОМ може да не се съдържат всичките ѝ компоненти. В този случай, на местата на липсващите компоненти се пише "*" и мрежата се нарича „редуцирана".

Статичната част на дадена обобщена мрежа се определя от елементите на множество $pr_{1,2,6,7} A$, т.е. от входните и изходните позиции на мрежата, от индексираната матрица с капацитетите на дъгите и от типа на преходите. Динамичният характер на мрежата се определя от ядрата на ОМ и от условията на преходите. Темпоралният характер се обуславя от времевите компоненти T, t°, t^* и от елементите на множество $pr_{3,4}$ A. Накрая, компонентите Φ, X *и* b играят ролята на памет на ОМ. Отделните функции също са свързани с тези четири компоненти на ОМ: функции π_A, π_L, c – със статичната структура; f, π_K – с динамичните елементи, а θ_1, θ_2 и θ_K – с времевите параметри.

1.1.2 Алгоритми за функциониране на преход и ОМ

В [19, 2, 22] са описани алгоритми за функциониране на прехода и цяла ОМ, които в годините бяха подобрени в [3, 4].

Общ алгоритъм за функциониране на преход в определен момент от време t_1

1. Входните позиции се подреждат по приоритет;

2. Всички ядра във входните позиции се подреждат по приоритет. Ядрата във входните позиции предварително се разделят на две групи – група на ядрата, които могат да извършат преход към изходна позиция, и група на ядрата, които не могат да извършат преход към изходна позиция;
3. Присвоява се стойност "0" (съответстваща на стойност "лъжа") на всички елементи на матрицата от предикатите *r* (условието на прехода), за които:
   - входна позиция, съответстваща на предиката, е празна;
   - изходна позиция, съответстваща на предиката, е пълна;
   - текущият капацитет на дъгата между входната и изходната позиция е нула, т.е. $m_{ij}=0$;
4. Изчисляват се останалите елементи на матрицата от предикати *r*;
5. Всички ядра с най-висок приоритет и стойност на предикатите "лъжа" се преместват в групата на ядрата, които не могат да извършат преход към изходна позиция. В тази група се преместват също и ядрата, за които предикатите имат стойност "истина", но междувременно изходните позиции са се запълнили от ядра с по-висок приоритет;
6. Според приоритета на входните позиции, от входните към изходните позиции на прехода се придвижват ядрата с най-висок приоритет;
7. Изчисляват се стойностите на характеристичната функция Φ за всеки преход;
8. Текущата стойност t' на моделното време се увеличава с t° ($t'=t_1+ t^\circ$);
9. Проверява се достигнато ли е време $t_1+ t_2$;
10. Ако отговорът на стъпка 9 е "не", се преминава към стъпка 2, ако е "да" – край на функционирането на прехода.

Общалгоритъм за функциониране на обобщената мрежа в определен момент от време t_T

1. В мрежата се въвежда всяко ядро α, за което $\theta_k(\alpha) \le T$;
2. Съставя се абстрактен преход на обобщената мрежа. Това е преход с динамична структура и представлява обединение на активните преходи в определен момент от време t_T;
3. Проверява се дали стойността на текущото време е по-малка от $T+t^*$;
4. Ако отговорът на стъпка 3 е "не", се отива към стъпка 12;
5. Определят се всички преходи, за които моментът за активиране t_1 е равен на текущия момент от време t_T;
6. Изчислява се булевият израз □ за всички преходи, определени на стъпка 5;
7. Към абстрактния преход се добавят всички преходи, за които булевите изрази, изчислени на стъпка 6, имат стойност "истина";
8. За абстрактния преход се прилага алгоритъм за функциониране на преход за една времева стъпка t°;
9. Всички преходи, на които е изтекло времето на активно състояние, се отстраняват от абстрактния преход;
10. Текущото време се увеличава с t° ($t_T = t_T + t^\circ$);
11. Преминава се към стъпка 2;
12. Край на функционирането на обобщената мрежа.

1.2 Въведение в невронните мрежи

1.2.1 Дефиниция на невронна мрежа

Работата върху изкуствени невронни мрежи, наричани накратко невронни мрежи, е мотивирана първоначално от разпознаването на човешкия мозък, който работи по коренно различен начин от конвенционалните цифрови изчислителни машини. Той има възможността да организира своите съставни части, наречени неврони, така че да извърши определени дейности, например разпознаване на образи, разбиране, контролиране на движения и т.н., много по-бързо от най-бързия съществуващ компютър.

По-общо изразено, невронната мрежа е машина, която е проектирана да моделира начина, по който мозъкът изпълнява конкретна задача или интересуваща ни функция. В настоящото въведение за невронните мрежи се обръща внимание на прилаганите в практиката работещи решения, които използват процес на обучение. За да постигнат добри резултати, невронните мрежи разгръщат масивна взаимосвързаност на елементарни изчислителни клетки, наречени неврони, или информационно-обработващи единици. Оттук се въвежда следната дефиниция на невронна мрежа, разглеждана като адаптивна машина:

Невронна мрежа е масивно разпаралелен процесор, изграден от елементарни обработващи единици, които имат естествена склонност да съхраняват знания, натрупани от опит, правейки ги налични за употреба. Тя наподобява мозъка в два аспекта:

1. *Знанието се придобива от мрежата в нейната околна среда чрез процес на обучение.*
2. *Отегляването на връзките, познати като синапсови тегла, се използва за да се съхранят придобитите знания.*

Процедурата, използвана да изпълни процеса на обучение, се нарича обучаващ алгоритъм. Неговата функция е да се модифицират синапсови тегла по организиран начин, за да се постигнат желаните изходни резултати. Модификацията на синапсови тегла предоставя традиционен метод за изграждане на невронните мрежи.

1.2.2 Основни свойства на невронните мрежи

Невронната мрежа притежава големи изчислителни възможности благодарение на масивната си паралелно разпределена структура и възможността да „учи" и следователно да обобщава обработената информация. Обобщаването се отнася до адекватния резултат на невронната мрежа на изходите при определени входни въздействия, които не са били използвани по време на обучението. Тези две информационно обработващи възможности правят възможно невронните мрежи да намират добри апроксимационни решения на комплексни проблеми, които са трудни за обучение. На практика обаче, невронните мрежи не могат да предоставят

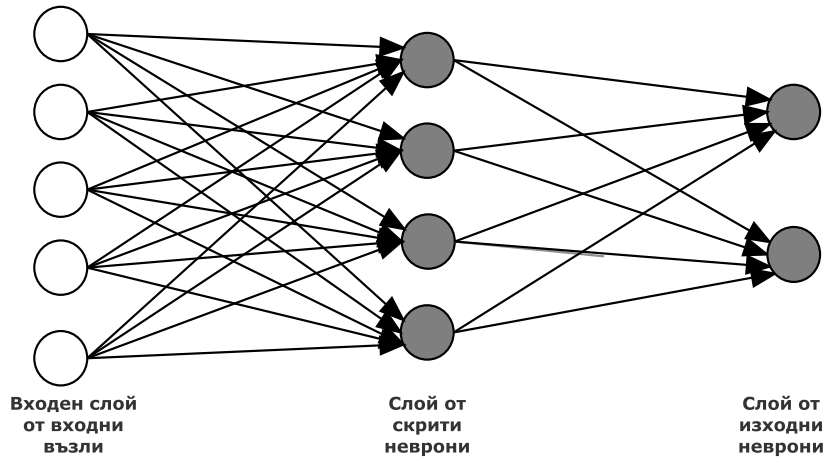
решение, работейки самостоятелно. По-скоро, те се нуждаят да бъдат интегрирани в цялостен системно-инженерен подход, а именно - един конкретен комплексен проблем да се декомпозира в определен брой относително прости задачи и на невронната мрежа да се присвои подмножество от задачите, които съответстват на техните присъщи възможности.

1.2.3 Мрежови архитектури

Начинът, по който невроните на една невронна мрежа се структурират, е тясно свързан с алгоритъма на обучение, използван за трениране на невронната мрежа. Следователно, може да се говори за алгоритъм на обучение, използван в дизайна на невронната мрежа, когато е била структурирана. Алгоритмите за обучение на някои от архитектурите ще бъдат разгледани по-долу в настоящия дисертационен труд.

1.2.3.2 Многослойни мрежи с право разпространение на сигнала

Класа от невронни мрежи с право разпространение на сигнала се отличава с наличие на един или повече скрити слоя, чиито изчислителни възли съответно са наречени скрити неврони. Терминът „скрит" се отнася за факта, че тази част на невронната мрежа не се вижда директно от входа или изхода на мрежата. Функцията на скритията неврон е да се намесва между външния вход и мрежовия изход по полезен начин. Чрез добавянето на един или повече скрити слоя мрежата е способна да извлече от изхода статистики от по-висок ред.



Фиг. 1.4. Пълно свързана мрежа с право разпространение на сигнала с един скрит слой и един изходен слой

Архитектурата на фиг. 1.4. илюстрира многослойна мрежа с право разпространение на сигнала за случай на един скрит слой. За простота мрежата на фиг. 1.4. се реферира като 5-4-2 мрежа, защото има 5 входни възела, 4 скрити неврона и 2 изходни неврона. Невронната мрежа на фиг. 1.4. е напълно свързана в смисъл, че всеки възел във всеки слой на мрежата е свързан към всеки друг възел в съседния последващ слой. Ако

някои от комуникационните връзки (синапсови връзки) липсват от мрежата, то тя се нарича частично свързана.

1.3 Невронни мрежи моделирани чрез обобщеномрежови модели

От дефинирането на обобщените мрежи през 1982 г. (първата статия [15] е публикувана през 1984 г.) до момента има публикувани над петдесет статии и три книги, в които са представени различни класове невронни мрежи, базови алгоритми за обучение и техни оптимизации с обобщеномрежови модели. За удобство в настоящата дисертация ще бъде използвано съкращението ОММ за обобщеномрежови модели. Класификацията на разгледаните публикации е направена според вида обучение – с учител и без учител. Подробности по видовете обучения и техните предимства са описани в [67]. Основен дял от статиите заемат ОММ на невронни мрежи, използващи обучение с учител. В тази група ОММ попадат деведнадесет статии [6, 21, 23, 40, 88, 89, 93, 96, 97, 101, 116, 133, 135, 138, 143, 145, 147, 152, 154], които използват многослойната архитектура с право разпространение на сигнала и модификации на алгоритъма с обратно разпространение на грешката [9, 73, 128].

Пет от статиите [21, 23, 100, 146, 153] представляват интерес с описаните нови разширения на алгоритмите за обучение на невронните мрежи, които позволяват паралелна оптимизация на алгоритмите с различни параметри на обучение. Първата статия за въвеждането на такъв нов оптимизационен подход е представена през 2007 г. в [21]. В нея авторът на дисертациония труд е докладвал за ОММ за паралелно обучение на дадена невронна мрежа с определени входове, изходи, набори за обучение и времево ограничение за обучението на невронната мрежа. Тази статия е основополагаща за серия от статии, в които се развива идеята от други автори за конкурентно и паралелно обучение на различни видове невронни мрежи.

През 2006г. екип от изследователи от Стандфордския университет, използвайки идеите, публикувани в [158, 159], създават софтуерна библиотека за паралелно изчисление на 10 алгоритъма за машинно обучение [36]. Между алгоритмите за машинно обучение попада и невронна мрежа с обратно разпространение на грешката. Тази разработка за паралелно обработване се базира на алгоритъма „Map-Reduce” [38]. Целите на разработката са да се използват ефективно многоядрените процесори с увеличаване броя на ядрата. Причината за преминаване към увеличаване броя на ядрата в съвременните процесори е опасността от огромна консумация на енергия от страна на процесорите над 2*GHz*, както и липса на материали, позволяващи непрекъснат растеж на тактовите честоти при работата на модерните компютри. Това е най-цитираната публикация, с над 780 цитирания към XII.2014, на тема машинно обучение и паралелна обработка. Тази разработка е подкрепена от DARPA и Intel, което показва сериозния интерес към тематиката [36]. Следва да се отбележи, че подходът за паралелно обучение на невронна мрежа с обратно

разпространение на грешката е близък с разработката, разглеждана по-долу в настоящия дисертационен труд.

За справка на вече публикувани статии е използвана и библиографията от [124, 8] и проучванията на автора на настоящия дисертационен труд по темата [1].

Глава 2. Обобщеномрежов модел на неврон

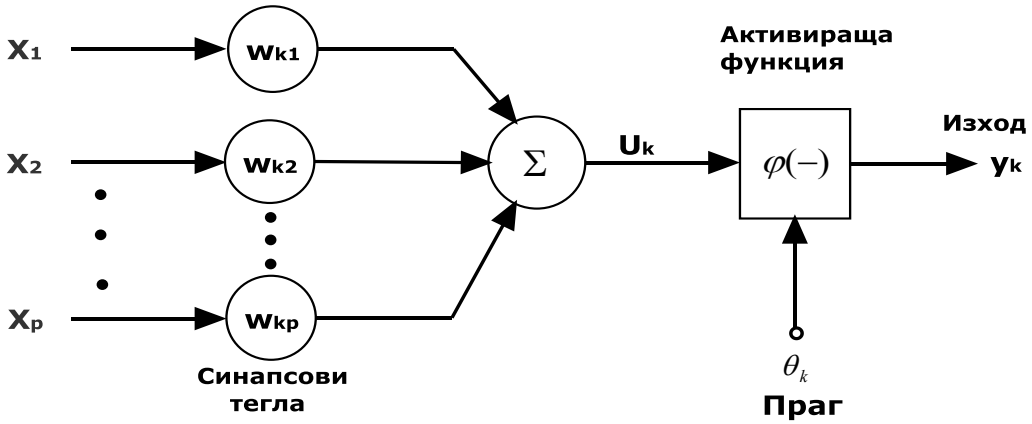
Цел на настоящата глава от дисертационния труд е да се направи математическо описание на основните неврони, използвани в невронните мрежи с право разпространение на сигнала, както и тяхното моделиране в обобщеномрежови модели, с цел по-нататъшна употреба при симулиране на невронни мрежи. Така поставената цел изпълнява задача 2 от задачите на дисертационния труд.

2.1 Математическо описание на модел на неврон

Работата върху изкуствени невронни мрежи, наричани за краткост само „невронни мрежи", е мотивирана от момента на потвърждението, че човешкият мозък работи (пресмята) по коренно различен начин от конвенционалните цифрови компютри. Усилията да бъде разучен човешкият мозък принадлежат до голяма степен на основополагащата работа на Рамон и Кахал (1911г.) [46], който въвежда идеята за неврона като структурна единица на мозъка.

Нека да бъде разгледана основната част от една невронна мрежа, каквато е невронът. Както е дефинирано в [65], един неврон е информационно-обработваща единица, която е фундаментална за оперирането на всяка невронна мрежа. На фиг. 2.1. е показан принципен изчислителен модел на неврон. Могат да бъдат идентифицирани три основни елемента от този невронен модел:

1. Едно множество от синапси или свързващи връзки, всяка от които се характеризира с тегло или сила сама по себе си. По-конкретно – един сигнал x_j на входа на синапса j, свързан към неврона k, е умножен със синапсовото тегло w_{kj}. Теглото w_{kj} е позитивно, ако асоцииранията синапс е възбуден, и отрицателно, ако синапсът е подтиснат.
2. Един суматор за сумиране на входните въздействия (сигнали), които са подтиснати или възбудени от тегловните коефициенти на синапсовите връзки на неврона. Операциите, описани тук, заместват един линеен комбинатор.
3. Една активираща функция за ограничаване на амплитудата на изхода всеки неврон. Активиращата функция също се цитира в литературата като „смачкваща" функция поради факта, че ограничава възможния амплитуден обхват на изходния сигнал до някаква крайна стойност. Обикновено, нормализираният амплитуден интервал на изхода на един неврон се записва като затворен единичен интервал [0,1] или алтернативно [-1,1].



Фиг. 2.1. Нелинеен модел на неврон

Моделът на неврон, показан на фиг. 2.1., също така включва външно приложен прагов коефициент θ_k, който има понижаващ ефект на мрежовия вход на активиращата функция. От друга страна, мрежовият вход на активиращата функция може да бъде повишен чрез употреба на отместващ член на активиращата фунцкия. Отместващият член на активиращата функция се явява противоположен на праговия коефициент θ_k.

Използвайки математически термини, един неврон k може да бъде описан чрез следната двойка уравнения:

$$Uk = \sum_{j=1}^{p} w_{kj} x_j \tag{2.1}$$

и

$$y_k = \varphi(u_k - \theta_k), \tag{2.2}$$

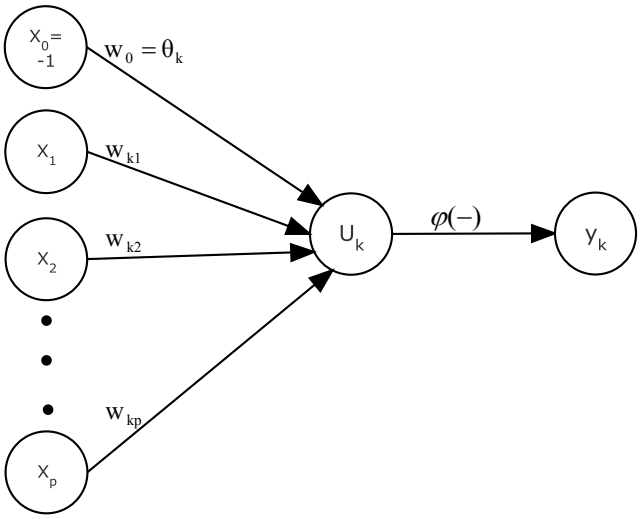
където x_1, x_2, … , x_p са входни сигнали; w_{k1}, w_{k2}, …, w_{kp} са синапсови тегла на неврона k; u_k е линейно комбиниращ изход; θ_k е прагова стойност; $\varphi(-)$ е активираща функция, а y_k е изходният сигнал на неврона.

Активиращата функция, отбелязана с $\varphi(-)$, дефинира изхода на неврона от гледна точка на активното ниво на неговия вход. Съществуват няколко вида активиращи функции, които са представени в [67].

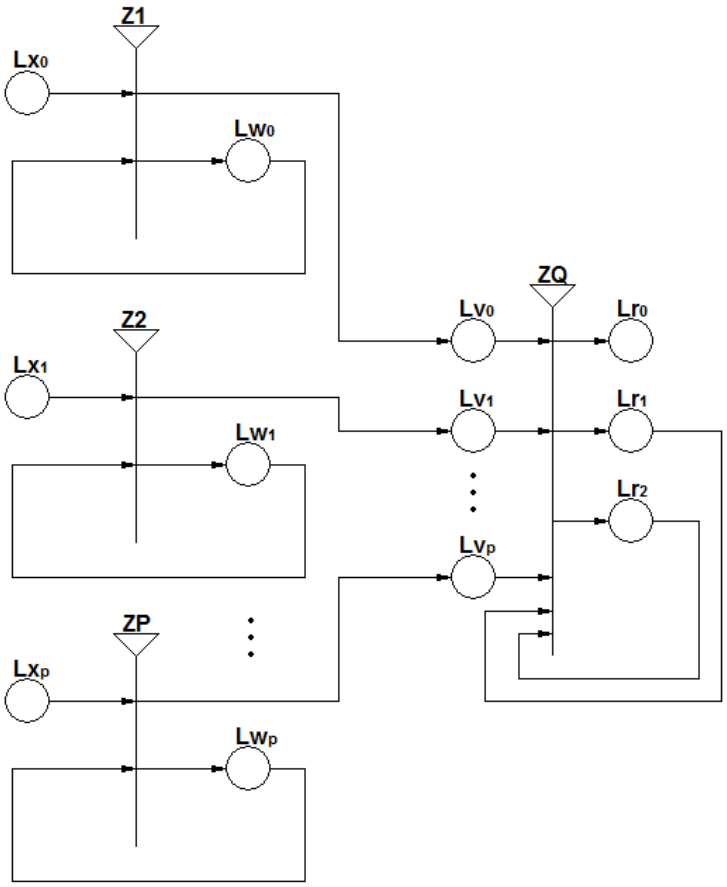
2.2 Обобщеномрежово представяне на модел на неврон

Обобщеномрежовото представяне може да бъде разделено на четири части – статична част, динамична част, времева част и памет. След като вече са разгледани и познати основните структури на обобщената мрежа, ще бъдат разделени важните части на модела на неврона на типичните за обобщените мрежи части. На първо място са поставени синапсите и тяхното представяне. Синапсите често биват представяни като ребра на сигнално-поточен граф с теглови коефициенти. Техният тип структура позволява да бъдат разглеждани като статична част на обобщена мрежа заедно с изхода на неврона. Всеки граф може да бъде представен чрез обобщена мрежа. Съществуват оператори, които могат да превърнат един граф в преходи на обобщена мрежа [83]. На фиг. 2.6 и фиг. 2.7 е показано представянето на

даден неврон като граф и също в статичната част на една обобщена мрежа.



Фиг. 2.6. Представяне на неврон като граф



Фиг. 2.7. Представяне на неврон чрез обобщена мрежа

Входните позиции L_{x0} … L_{xp} на преходите $Z1$ до ZP са входните връзки към неврона. Минавайки през някои от преходите $Z1$ до ZP, всяко ядро получава нова стойност от произведението $x_i.w_i$, където x_i е входен сигнал, който влиза на позиция L_{xi}. Всеки математически модел на неврон притежава суматор за всички входни сигнали, което е показано във формула (2.1). Суматорът може да бъде представен чрез статична част в обобщената мрежа на фиг. 2.7. Пример за това е даден за позиция Lr_2 и нейната

характеристична функция $\sum_{i=0}^{p} f(x_i, w_{ki})$ в матрицата на преходите r_1 (вж. фиг. 2.8). Тук трябва да се обърне внимание, че праговата стойност е маркирана като $x_0 = \theta$ и $w_{k0} = -1$. След представяне на суматора е необходимо да се разгледа и активиращата функция $\varphi\left(\sum_{i=0}^{p} f(x_i, w_{ki})\right)$, която е поставена на позиция Lr_1 на фиг. 2.8 в матрицата на преходите. Активиращата функция може да бъде прагова функция, линейно-отсечкова апроксимационна функция, сигмоидална функция и функция на Грийн[14] при еднакви теглови коефициенти със стойност 1. В този модел не се разглежда изчислението на функциите. Обект на интерес е процесът на обучение на неврона и неговото представяне. Преходите *Z*1 … *ZP* изпълняват същите операции върху входните сигнали, така че ще бъде разгледан само преходът *Z*1. Когато едно ядро влиза в позиция L_{x0}, една характеристична функция *SetChange* проверява дали характеристиката *Change* на влизащото ядро е *истина*. Ако *Change* е *истина*, тогава се взема характеристиката *Weight* на ядро W_0.

r_1	Lr_0	Lr_1	Lr_2
L_{x0}	False	False	True
L_{x1}	False	False	True
…	…	…	…
L_{xp}	False	False	True
Lr_1	$\varphi\left(\sum_{i=0}^{p} f(x_i, w_{ki})\right) \geq 0$	False	False
Lr_2	False	$\sum_{i=0}^{p} f(x_i, w_{ki}) \geq 0$	False

Ядро W_0 е постоянно разположено на позиция L_{w0} и съдържа теглото на синапс на неврона. Характеристиката *Change* се използва, защото всеки неврон има режим на обучение и има нужда за промяна на теглото на неврона. Когато *Change* е състояние "false", невронът е в класифициращ режим.

*Z*1	I_{wo}	I'_{o}
I'_{xo}	SetChange	SetChange
L_{wo}	False	False

*Фиг. 2.9. Матрица на преходите в Z*1

Тогава се умножава характеристиката *Value* от входното ядро с W_0. Резултатът е присвоен обратно на характеристиката *Value*. Сега ядрото може да бъде преместено в позиция L_{v0}. На фиг. 2.9 е показана матрицата на прехода *Z*1. Представеният ОММ е симулиран върху обобщеномрежов симулатор, създаден от Трифонов и Георгиев [160]. Резултатите от симулацията показват нормална работа на ОММ в сравнение с модел на изкуствен неврон. Предимствата на ОММ са конкурентната обработка на входните сигнали и възможността чрез друга характеристична функция на мястото на $\sum_{i=0}^{p} f(x_i, w_{ki}) \geq 0$ да бъдат изследвани решения с времезакъснения или спиране на изчислението на сумата от входни

въздействия на определена прагова стойност, когато е нужна по-бърза реакция на насищане от неврона и асинхронност на изкуствения неврон.

2.3 Постигнати резултати

В настоящата глава беше представен математически модел на неврон използван при мрежи с право разпространение на сигнала с четири различни преходни функции, част от които се използват в най-често използваните невронни мрежи. На база на представения математически модел на неврон беше представен работещ обобщеномрежов модел на неврон, който позволява конкурентна обработка на входните въздействия върху обобщеномрежовият модел на неврон от типа SIMD [47], както и възможността да се въведе допълнителна логика за намаляване броя на пресмятанията в дадения неврон при достигане на определен праг, след който невронът е вече активен. Обобщеномрежовият модел от тази глава е публикуван в [11].

Глава 3. Обобщеномрежов модел за паралелна оптимизация на многослоен перцептрон с обратно разпространение на грешката

Целта на настоящата глава от дисертационния труд е да представи обобщеномрежов модел на алгоритъм за конкурентна оптимизация на многослоен перцептрон с обратно разпространение на грешката. Така поставената цел изпълнява задачи 3 и 4 от задачите на дисертационния труд.

ОММ за конкурентна оптимизация на невронна мрежа с обратно разпространение на грешката, който се разглежда в настоящата глава от дисертацията, е разработен и публикуван през 2007 г. [21]. Негова важна характеристика е употребата на алгоритъма на *златното сечение* [41] при избора на оптимални параметри за паралелното обучение на невронни мрежи. Описанието на работата на модела е направено чрез апарата на обобщените мрежи, който позволява моделиране на паралелни математически процеси. Представената невронна мрежа е многослоен перцептрон с обратно разпространение на грешката, която ще бъде разгледана в т. 3.2.

3.1 Алгоритъм на златното сечение

Алгоритъмът на *златното сечение*, който е използван в гореспоменатия модел, се базира на разработката на Дънлап за определяне на интервала за търсене на оптималния брой скрити неврони в невронната мрежа при крайно време за нейното обучение [41]. Тук ще бъде дефиниран алгоритъмът на *златното сечение* спрямо броя на скритите неврони в междинния слой на многослойния перцептрон, който ще бъде оптимизиран.

Нека са дадени естественото число N и реалното число C. Те съответстват на максималния брой скрити неврони и минималната желана грешка. Нека реалната монотонна функция f определя грешката $f(k)$ на

невронната мрежа с k скрити неврона. Нека функцията $R \times R \to R$ бъде дефинирана за всяко $x, y \in R$ по следния начин:

$$c(x,y) = \begin{cases} 0; \text{ ако } \max(x,y) < C \\ \frac{1}{2}; \text{ ако } x \leq C \leq y \\ 1; \text{ ако } \min(x,y) > C \end{cases}.$$

Нека $\varphi = \frac{\sqrt{5}+1}{2} = 0.61\ldots$ е коефициентът на златното сечение. Нека първоначално се инициализират:

$L = 1$; $M = [\varphi^2\text{:N}]+1$, където $[x]$ е цялата част на реално число $\text{x} \geq 0$.

Стъпките на алгоритъма са следните:

1. Ако $L \geq M$, отиди на стъпка 5.
2. Изчисли *c(f(L), f(M))*. Ако
$$c(x,y) = \begin{cases} 1, \text{отиди на стъпка } 3 \\ \frac{1}{2}, \text{отиди на стъпка } 4 \\ 0, \text{отиди на стъпка } 5 \end{cases}$$
3. $L = M + 1$; $M = M + [\varphi^2.(N-M)] + 1$, отиди на стъпка 1.
4. $M = L + [\varphi^2.(N-M)] + 1$; $L = L + 1$, отиди на стъпка 1.
5. Край, резултатът от работата на алгоритъма е L.

3.2 Многослоен перцептрон

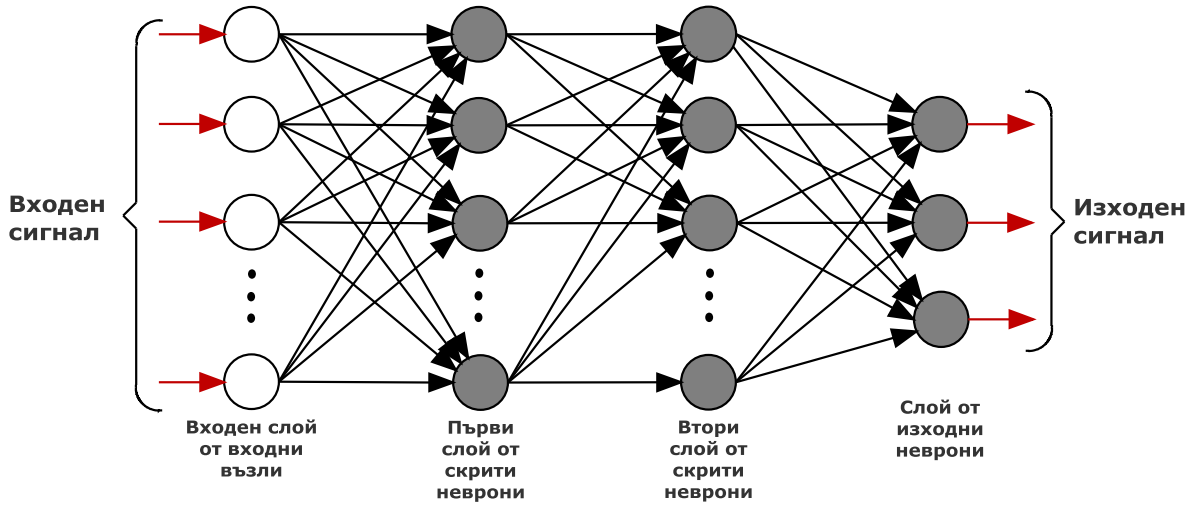
Многослойният перцептрон представлява многослойна невронна мрежа с право разпространение на сигнала. Може да се прилага за класифициране на нелинейни разделими образи. Нелинейната характеристика на образи не е възможно да се реализира с еднослойна невронна мрежа с перцептрон на Розенблат и адаптивно филтриране, използвайки алгоритъма „Метод на най-малките квадрати“ [67]. Популярен метод за обучение на многослоен перцептрон е алгоритъмът за обратно разпространение на грешката, който включва „Метод на най-малките квадрати“ [25, 112] като специален случай. Процедурата за обучение се изпълнява в две фази:

1. В правата фаза на обучение синапсовите тегла на мрежата са фиксирани, а входният сигнал се разпространява през мрежата слой по слой, докато достигне изходния слой. По този начин в тази фаза промените са ограничени до активиращи потенциали и изходи на невроните в мрежата.
2. В обратната фаза един сигнал на грешката се генерира, сравнявайки изходите на мрежата с желания отговор. Резултатният сигнал на грешката се разпространява по мрежата отново слой по слой, но този път разпространението е изпълнено в обратна посока. В тази втора фаза последващи корекции се правят към синапсовите тегла на мрежата. Изчисленията на корекциите за изходния слой са разбираеми, но са по-трудни за скритите слоеве.

Разработката на алгоритъма за обратно разпространение на грешката в средата на 80-те години на XXв. представлява повратна точка в областта на невронните мрежи с това, че те предоставят изчислимо ефективен метод за обучение на многослойни перцептрони. По този начин отшумява песимизмът относно обучението в многослойните перцептрони, който е загатнат в [112].

3.2.1 Въведение в многослойния перцептрон

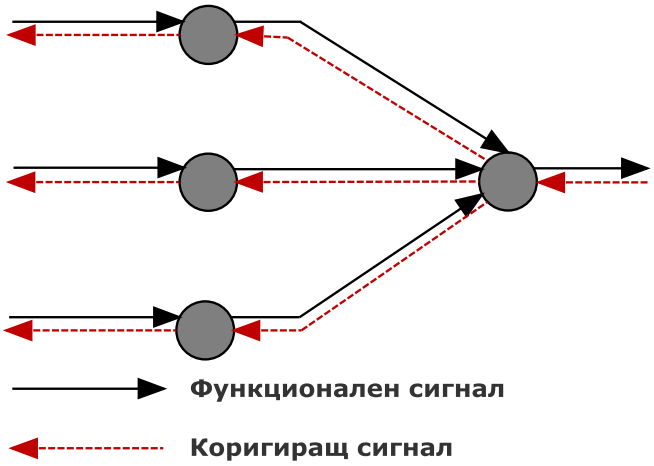
На фиг. 3.1. е показана архитектура на многослоен перцептрон с два скрити и един изходен слой. За да бъде представено описанието на многослойния перцептрон в неговата обща форма, показаната мрежа е пълно свързана. Това означава, че всеки неврон във всеки слой на мрежата е свързан към всички неврони (възли) на предишния слой.



Фиг. 3.1. Архитектурен граф на многослоен перцептрон с два скрити слоя

Сигналът в мрежата се разпространява в права посока, от ляво на дясно и слой по слой. На фиг. 3.2. е илюстрирана част от многослойния неврон. Два вида сигнали са идентифицирани в тази мрежа:

1. **Функционални сигнали.** Функционален сигнал е входен сигнал, който идва на входа на мрежата, разпространява се през нея и се появява на изхода ѝ като изходен сигнал. Причините да бъде наречен „функционален сигнал" са две. Първо, той е предвиден да изпълни полезна дейност на изхода на мрежата. Второ, за всеки неврон на мрежата, през който един функционален сигнал премине, сигналът се изчислява от входовете и асоциираните тегла, приложени към този неврон. Функционалният сигнал също се дефинира и като входен сигнал.
2. **Коригиращи сигнали.** Един коригиращ сигнал произхожда от един изходен неврон на мрежата и се разпространява назад (слой по слой) през мрежата. Той се нарича коригиращ, защото неговото изчисление за всеки неврон от мрежата включва коригиращо-зависима функция в една или друга форма.



Фиг. 3.2. Илюстрация на посоките на двата основни потока в многослойния перцептрон – право разпространение на функционален сигнал и обратно разпространение на коригиращ сигнал

3.2.1.1 Функция на скритите неврони

Скритите неврони действат като детектори на характеристики. Като такива, те играят критична роля в операциите на многослойния перцептрон. За повече подробности виж [67].

3.2.2.5 Кратко представяне на алгоритъма с обратно разпространение на грешката

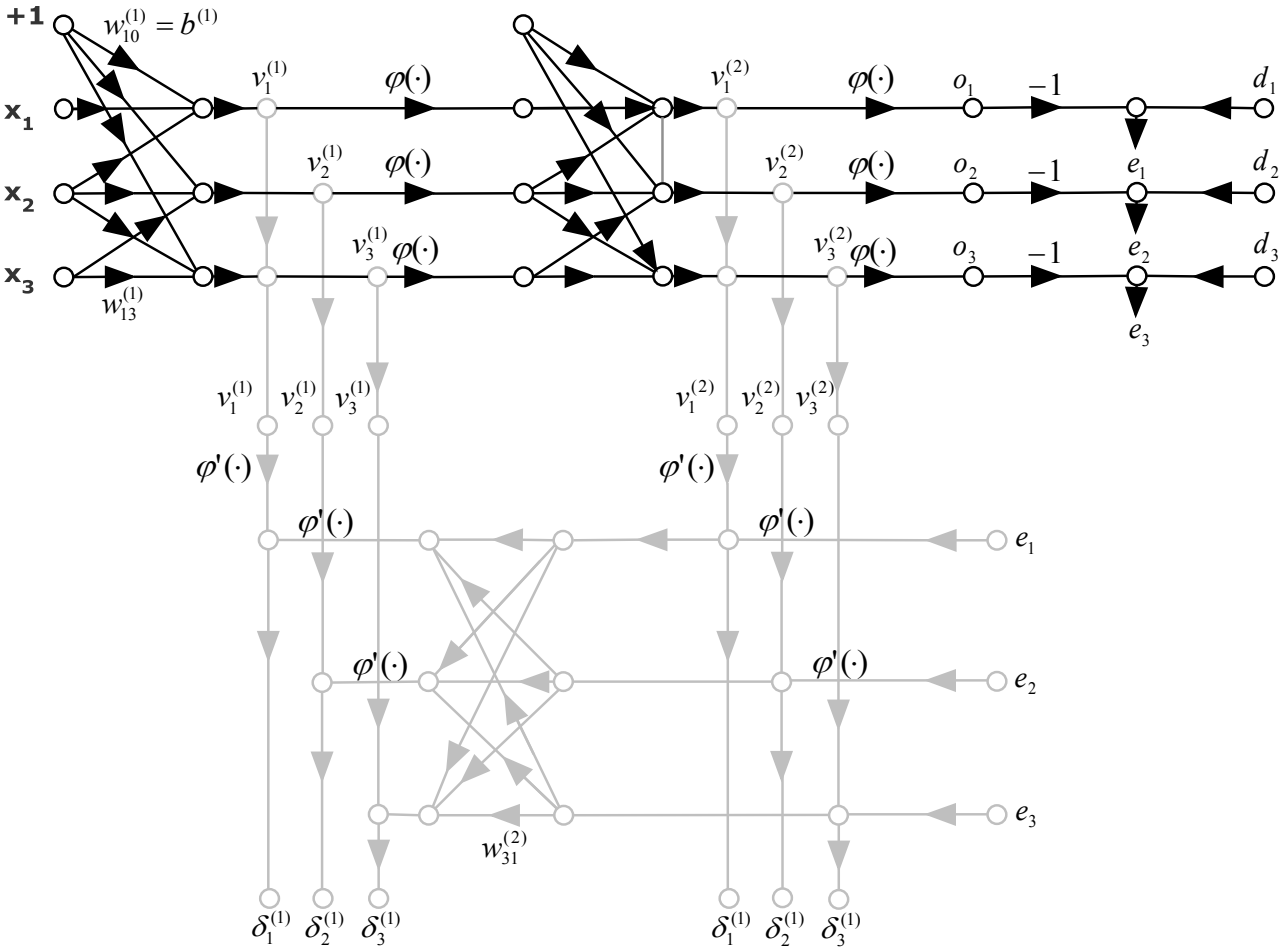
На фиг. 3.1 е представена мрежова архитектура на многослоен перцептрон. Съответният сигнално-поточен граф за обратно разпространяващо обучение, показващ съответно смесването едновременно на правата и обратната фаза на изчисления, включени в процеса на обучение, е представен на фиг. 3.7 за случая на $L=2$ и $m_0=m_1=m_2=3$. Най-горната част на сигнално-поточния граф отразява правата фаза, а долната част – обратната фаза, като втората се посочва като *граф на чувствителността* за изчисляване на локалните градиенти в алгоритъма за обратно разпространение на грешката [112].

1. **Представяне на наборите за обучение.** Предоставят се наборите за обучение на невронната мрежа. За всеки набор, подредени по някакъв начин, се изпълнява последователността за изчисляване на правия и обратния ход на обучение, описани по-долу в точка **2** и **3**.

2. **Изчисляване на правия ход на обучение.** Нека наборът за обучение в епохата да е отбелязан с ($\boldsymbol{x}(n)$, $\boldsymbol{d}(n)$), с входен вектор $\boldsymbol{x}(n)$, приложен към входния слой от възли, и желаният отговор векторът $\boldsymbol{d}(n)$, представен на изходния слой от неврони. Изчисляват се индуцираните локални въздействия и стойностите на активиращите функции през мрежата напред, продължавайки слой по слой. Индуцираното локално въздействие $v_j^{(l)}(n)$ за неврона j в слоя l е:

$$v_j^{(l)}(n) = \sum_i w_{ji}^{(l)}(n) y_i^{(l-1)}(n), \qquad (3.39)$$

където $y_i^{(l-1)}(n)$ е изходната стойност на неврона *i* в предишния слой *l-1 на n-тата итерация, а* $w_{ji}^{(l)}(n)$ е синапсовото тегло на неврона *j* в слоя *l, на който е подадена стойност от неврона i* в слоя *l*–1. За *i = 0* се получава $y_0^{(l-1)}(n) = +1$, а $w_{j0}^{(l)}(n) = b_j^{(l)}(n)$ е отместването, приложено към неврона *j* в слоя *l.*



Фиг. 3.7. Сигнално-поточен граф, представящ алгоритъма за обратно разпространение на грешката. Горна част на графа – прав ход, долна част – обратен ход.

Ако се приеме, че се използва сигмоидална функция, изходната стойност на неврона *j* в слоя *l* е

$$y_j^{(l)} = \varphi_j(v_j(n)).$$

Ако невронът *j* е в първия скрит слой (т.е. *l=1*), да се използва

$$y_j^{(0)}(n) = x_j(n),$$

където $x_j(n)$ е *j*-тият елемент на входния вектор *x(n)*.
Ако невронът *j* е в изходния слой (т.е. *l=L*, където *L* се използва

като дълбочина на мрежата) да се използва

$$y_j^{(L)} = o_j(n).$$

Изчислява се грешката:

$$e_j(n) = d_j(n) - o_j(n), \tag{3.40}$$

където $d_j(n)$ е j-тият елемент на желания отговор от вектора ***d****(n)*.

3. **Изчисляване на обратния ход на обучение.** Изчисляват се δ-те на мрежата, дефинирани по следния начин

$$\delta_j^{(l)}(n) = \begin{cases} e_j^{(L)}(n)\varphi_j'\left(v_j^{(L)}(n)\right) & \text{за неврона } j \text{ в изходния слой } L \\ \varphi_j'(v_j^{(l)}(n))\sum_k \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n) & \text{за неврона } j \text{ в скрития слой } l \end{cases}, \tag{3.41}$$

където производната $\varphi_j'(\cdot)$ отбелязва диференцирането по отношение на аргумента. Коригират се синапсовите тегла на мрежата в слоя *l* според общото делта правило

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha\left[w_{ji}^{(l)}(n-1)\right] + \eta\delta_j^{(l)}(n) y_i^{(l-1)}(n), \tag{3.42}$$

където η е степента на обучение, а α е моментната константа.

4. **Повторение.** Повтарят се изчисленията в правия и обратния ход от точка **2** и **3** чрез представяне на нови епохи на обучителни набори към мрежата, докато избраният критерий за прекратяване на обучението е достигнат.

3.3 Представяне на ОММ за паралелна оптимизация на невронна мрежа с обратно разпространение на грешката

Обобщеномрежовият модел, показан на фиг. 3.8., е редуциран. Той не притежава темпорални компоненти, приоритети на преходите и няма ограничения в капацитетите на позициите и дъгите.

ОМ моделът се състои от пет прехода:

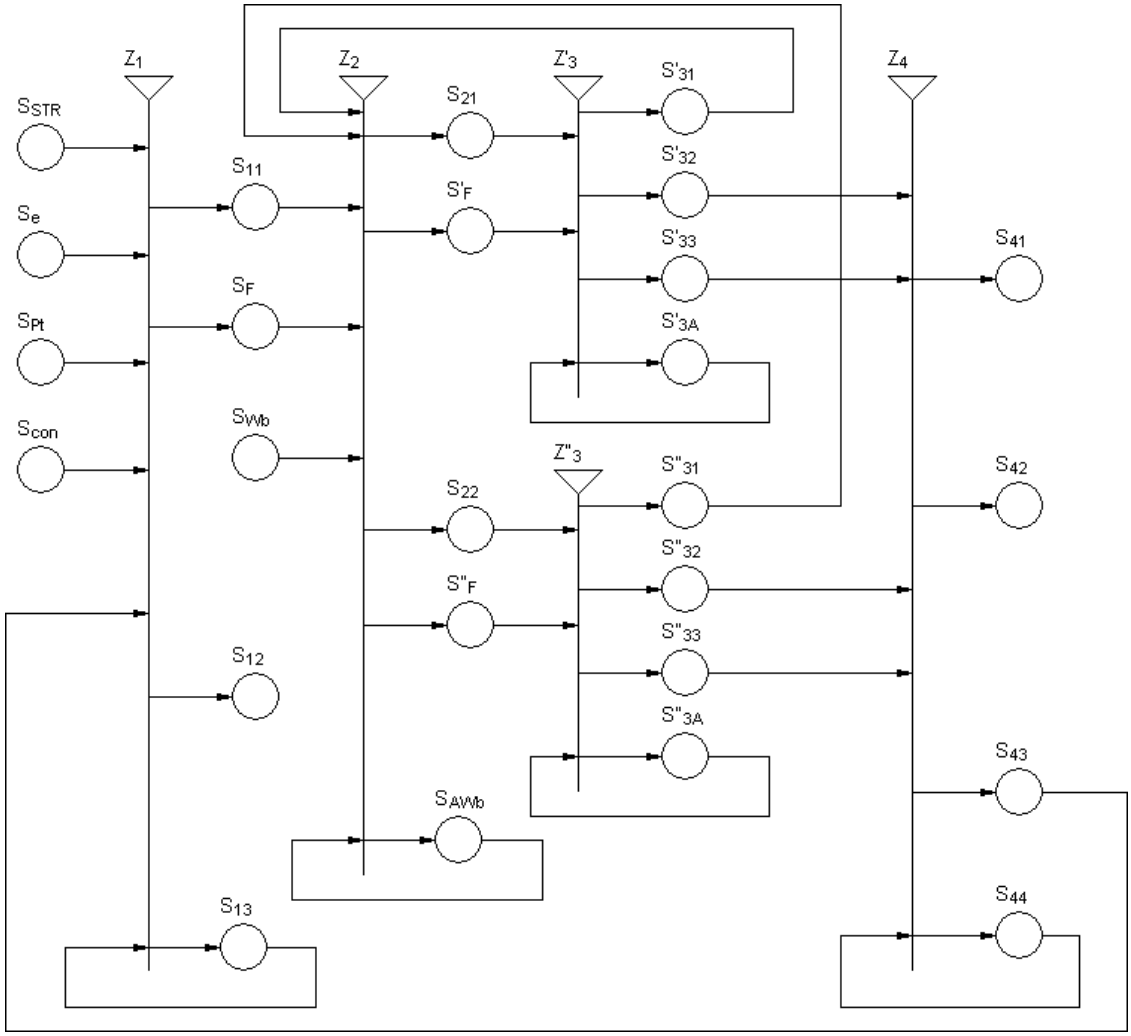
$$A = \{Z_1, Z_2, Z'_3, Z''_3, Z_4\},$$

където преходите описват следните процеси:

- Z_1 – Формиране на начални условия и структура на невронните мрежи;
- Z_2 – Подготвяне на ядра с необходимата информация за обучение на две невронни мрежи едновременно;
- Z'_3 – Изчисляване на правия и обратния ход на многослойния перцептрон за първата мрежа;
- Z''_3 – Изчисляване на правия и обратния ход на многослойния перцептрон за втората мрежа;

Z_4 – Проверка на резултатите от обучението на невронните мрежи и взимане на решение да продължи или да спре процесът на обучение.

Алгоритъма на работа на прехода Z''_3 е еднакъв с този на Z'_3, и няма нужа да бъде разглеждан детайлно.

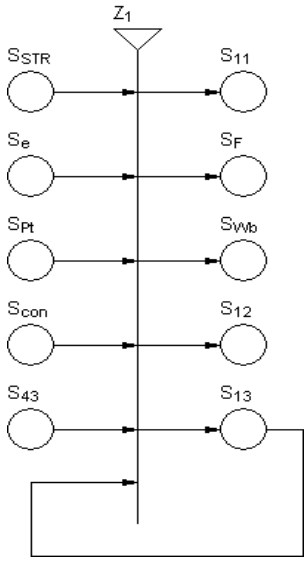


Фиг. 3.8. Обобщеномрежов модел за паралелна оптимизация на невронна мрежа с обратно разпространение на грешката

3.3.1 Описание на преход Z_1

$$Z_1=<\{S_{str}, S_e, S_{Pt}, S_{con}, S_{43}, S_{13}\}, \{S_{11}, S_F, S_{wb}, S_{12}, S_{13}\}, R_1,$$
$$\wedge(\vee(\wedge(S_e, S_{Pt}, S_{con}), S_{13}), \vee (S_{STR}, S_{43})) >$$

Преходът Z_1 се използва като входна точка за всички ядра, носещи информация за основните параметри на невронната мрежа. Всеки преход има условие на прехода, което определя активирането му. За прехода Z_1 то е $\wedge(\vee(\wedge(S_e, S_{Pt}, S_{con}), S_{13}), \vee (S_{STR}, S_{43}))$. От условието става ясно, че задължителни активни позиции, от които зависи активирането на прехода, са S_{STR} или S_{43} и S_{13} или всички позиции S_e, S_{Pt}, S_{con} едновременно. Под активна позиция се има предвид входна или изходна позиция, в която е постъпило ядро. В позиция S_{STR} постъпва ядро *a*, което носи две характеристики.



Фиг. 3.9. Графично представяне на прехода Z_1

Първата характеристика е броят неврони във входния слой, а втората характеристика е броят неврони в изходния слой. Следващата входната позиция е S_e, в която постъпва ядро β. То съхранява максималната допустима средна квадратична грешка E_{max} в характеристиката x_0^{β} и брой епохи на обучение, съхранени в характеристиката x_1^{β}. Тези характеристики служат като условия за прекратяване процеса на обучение на дадена невронна мрежа. S_{Pt} е входна позиция, в която постъпва ядрото γ. Ядрото γ има за характеристика всички входни и изходни набори за обучение на невронната мрежа. Позицията S_{con} приема ядро ξ, което притежава характеристика x_0^{ξ} – максимален брой скрити неврони в скрития слой. В позиция S_{43} на по-късен етап постъпват ядра θ и δ, които служат за иницииране на обучение с променен интервал на *златното сечение* от прехода Z_4. Позиция S_{13} проверява началните условия за алгоритъма на з*латното сечение* преди изпълнението му. На табл. 3.1. е представена таблица с пълното описание на всяко ядро с началните му характеристики и началната позиция, през която ядрото постъпва в прехода Z_1. Преминаването на ядрата към изходните позиции $<S_{11}, S_F, S_{12}, S_{13}>$ в прехода Z_1 зависи от матрицата на преходите R_1, представена на фиг. 3.10 по-долу.

Ядро	Начални характеристики	Позиция на създаване
α	x_0^{α}, x_1^{α}	S_{str}
β	x_0^{β},x_1^{β}	S_e
γ	x_0^{γ}, x_1^{γ}	S_{pt}
δ	x_0^{δ}	S_{43}
ξ	x_0^{ξ}	S_{con}
θ	x_0^{θ}, x_1^{θ}, x_2^{θ},x_3^{θ}, x_4^{θ}	S_{11}/S_{43}

Табл. 3.1. Таблица на ядрата в преход Z_1

R_1	S_{11}	S_F	S_{12}	S_{13}
S_{STR}	False	False	False	True
S_e	False	False	False	True
S_{Pt}	False	False	False	True
S_{con}	False	False	False	True
S_{43}	$W_{S43,S11}$	$W_{S43,SF}$	False	False
S_{13}	$W_{13,11}$	False	$W_{13,12}$	True

Фиг. 3.10. Матрица на предикатите R_1

Повечето преминавания на ядрата са директно разрешени или забранени от входните към изходните позиции, както е указано в R_1. В случаите, в които е необходимо да се проверяват допълнителни условия за преминаването от позиция S_{13} към позиция S_{11}, се използва предикатът $W_{13,11}$. Условието на предиката за преминаване на ядро към изходната позиция S_{11} е дали интервалът на *златното сечение* [*L*, *M*] е разделим. Този интервал е предоставен от характеристика x_1^{θ} на ядрото *θ* или от ядрото *i*, постъпило във входна позиция S_{43}, и неговата характеристиката x_0^{i}, съдържаща интервала [*L*, *M*]. Интервалът [*L*, *M*] представлява текущия работен интервал на алгоритъма за *златно сечение* с долна граница минималният брой скрити неврони и с горна граница максималният брой такива. Предикатите $W_{S43,S11}$ и $W_{S43,SF}$ служат за разпределяне на ядрата *θ* и *δ* към позиции S_{11} и S_F при постъпване от прехода Z_4. В табл. 3.3 по-долу са описани предикатите на прехода Z_1.

Предикат	Описание
$W_{13,11}$	Не е възможно да се раздели интервалът [*L*, *M*]
$W_{13,12}$	Възможно да се раздели интервалът [*L*, *M*]
$W_{S43,S11}$	Ако името на ядрото е *θ* на позиция S_{43}
$W_{S43,SF}$	Ако името на ядрото е *δ* на позиция S_{43}

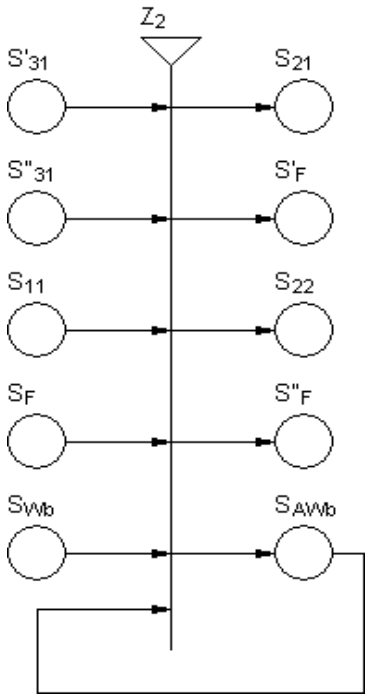
Табл. 3.3. Описание на предикатите в R_1

На входната позиция S_{13} се активира характеристична функция *CharS13S11GenerateTetaToken*. Необходимото и достатъчно условие за активиране на функцията е постъпването на ядрата *α*, *β*, *γ* и *ξ*. От получената информация от посочените по-горе ядра се генерира ядрото *θ*, което преминава на изходната позиция S_{11}. Ядрото *θ* служи за пренос на необходимата информация за структурата на невронната мрежа. По този начин целта на прехода Z_1 да подготви структурата на невронната мрежа и интервала на *Златното Сечение*, в който ще се търсят най-добре обучени невронни мрежи в конкурентни преходи, е постигната.

3.3.2 Описание на преход Z_2

$$Z_2=<\{S'_{31}, S''_{31}, S_{11}, S_F, S_{Wb}, S_{AWb}\}, \{S_{21}, S'_F, S_{22}, S''_F, S_{AWb}\}, R_2,$$
$$\vee(\wedge(S_F, S_{11}),\ \vee(S_{AWb}, S_{Wb}),\ \vee(S'_{31},\ S''_{31}))>$$

Преходът Z_2 служи за изграждането на две отделни невронни мрежи с необходимата информация, които ще бъдат обучавани от следващите два прехода Z'_3 и Z''_3.



Фиг. 3.11. Графично представяне на прехода Z_2

Условието за активиране на прехода е $\vee(\wedge(S_F, S_{11}), \vee(S_{AWb}, S_{Wb}), \vee(S'_{31}, S"_{31}))$. Първоначално на позиция S_{11} се генерира ядро θ от характеристична функция в прехода Z_1. Ядрото θ има пет характеристики x_0^θ, x_1^θ, x_2^θ, x_3^θ и x_4^θ. Тези характеристики описват структурата на невронната мрежа и са представени в табл. 3.4. На позиция S_F постъпва ядро δ, което носи описание на преходните функции *f*1, *f*2 и *f*3 за входния слой, скрития слой и за изходния слой. Така описаните функции *f*1, *f*2 и *f*3 за всеки слой на невронната мрежа улесняват пресмятането и в преходите Z'_3 и Z''_3. Позиция S_{Wb} приема ядро ε, което има характеристики, съхраняващи матрицата с теглата на невронната мрежа *w* и вектор *b* с отместващите коефициенти за всеки скрит неврон. Първоначално при активиране на прехода във входната позиция S_{AWb} няма постъпващо ядро, а на по-късен етап минават ядра, постъпващи от преходите Z'_3 и Z''_3. В табл. 3.4. е представено подробно описание на ядрата с техните характеристики и позиция на създаване.

Ядро	Начални характеристики	Позиция на създаване
$\boldsymbol{\theta}$	$x_0^\theta, x_1^\theta, x_2^\theta, x_3^\theta, x_4^\theta, x_5^\theta, x_6^\theta$	S_{11}
$\boldsymbol{\delta}$	x_0^δ	S_F
$\boldsymbol{\varepsilon}$	x_0^ε, x_1^ε	S_{Wb}

Табл. 3.4. Таблица на ядрата в преход Z_2

Преди да преминат ядрата от входните към изходните позиции се изпълняват някои характеристични функции, които са описани по-детайлно в табл. 3.6 по-долу.

Предусловие	Позиция на стартиране	Позиция на завършване
CharSFDeltaSplit	S_F	S'_F, S''_F
CharS11TetaSplit	S_{11}	S_{21}, S_{22}

Табл. 3.6. Таблица на характеристичните функции на прехода Z_2

Първата характеристична функция, която се изпълнява е *CharSFDeltaSplit*. Целта на тази функция е да раздели ядрото *δ* на *δ*‘ и *δ*“. Ядрата *δ*‘ и *δ*“ са копия на ядрото *δ*. Тези две ядра съответно преминават към позиции S'_F и S''_F. При постъпване на ядрото *θ* в позиция S_{11} и на на ядро *ε* в позиция S_{Wb} характеристичната функция *CharS11TetaSplit* се активира. Функцията *CharS11TetaSplit* разделя ядрото *θ* на ядрата *θ*‘ и *θ*“ и добавя четири нови характеристики $x_5^\theta, x_6^\theta, x_7^\theta$ и x_8^θ във всяко ядро. Характеристиката x_5^θ е масив, съхраняващ резултата на изходите от правия ход на обучение, а x_6^θ пази степента на обучение за дадена невронна мрежа. Характеристиките x_7^θ и x_8^θ са копия на характеристиките x_0^ε и x_1^ε на ядрото *ε*. Така създадени ядра *θ′* и *θ″* носят структурната информация за невронната мрежа за всеки от преходите Z'_3 и Z''_3, в които ще се извършва обучението.

Разрешаването за преминаване от входни към изходни позиции на ядрата се извършва при определени условия, които са дефинирани в матрицата на преходите R_2, която е представена на фиг. 3.12.

R_2	S_{21}	S'_F	S_{22}	S''_F	S_{AWb}
S'_{31}	$W_{S'31,S21}$	$W_{S'31,S'F}$	False	False	False
S''_{32}	False	False	$W_{S''31,S22}$	$W_{S''31,S''F}$	False
S_{11}	$W_{S11,S21}$	False	$W_{S11,S22}$	False	False
S_F	False	$W_{SF,S'F}$	False	$W_{SF,S''F}$	False
S_{Wb}	False	False	False	False	False
S_{AWb}	True	False	True	False	False

Фиг. 3.12. Матрица на предикатите R_2

В повечето случаи преминаването или забраната за преминаване от входните към изходните позиции не носи никаква информация. Интересни са случаите за дефиниране на предикатите $W_{S11,S21}$ и $W_{S11,S22}$ при преминаването от входната позиция S_{11} на ядра *θ*‘ към позиция S_{21} и на ядро *θ*“ към позиция S_{22}. Предикатът $W_{S11,S21}$ разрешава преминаването на *θ*‘ към позиция S_{21}, ако ядрото се намира в позиция S_{11}. По същия начин се прави проверка и за ядрото *θ*“ от позиция S_{11} към позиция S_{22} в предиката $W_{S11,S22}$. Причината да се разделят и разпределят ядрата към преходи Z'_3 и Z''_3 е възможността за конкурентна работа на тези преходи с характеристиките за описание на невронната мрежа. Предикатът $W_{SF,S'F}$ разрешава преминаването на ядро *δ*‘ от позиция S_F към позиция S'_F. По същия начин сработва предикатът $W_{SF,S''F}$ за ядрото *δ*“. Целта на разделянето и разпределянето на ядрата *δ*‘ и *δ*“ е да дадат възможност за доставяне и конкурентно използване на преходните функции *f*1, *f*2 и *f*3 за всеки слой на невронните мрежи, които ще бъдат изчислявани едновременно. Предикатите $W_{S'31,S21}$, $W_{S'31,S'F}$, $W_{S''31,S22}$ и $W_{S''31,S''F}$ са от значение при

обучението на невронните мрежи в преходите Z'_3 и Z''_3. След първоначалното обучение и преминаване през преходите Z'_3 и Z''_3, преходите се активират отново при постъпване на конкретни ядра на определени позиции, за да се продължи обучението на невронните мрежи. Начинът за реактивиране на обучението за прехода Z'_3 е постъпването на ядрата θ' и δ' в изходните позиции S_{21} и S'_F. За да се случи това, при постъпване на двете ядра от прехода Z'_3 в позиция S'_{31}, която е обща за двата прехода, се активират предикатите $W_{S'31,S21}$ и $W_{S'31,S'F}$. Преходът $W_{S'31,S21}$ разрешава преминаването на ядрото θ' от входната позиция S'_{31} в позиция S_{21}. $W_{S'31,S'F}$ разрешава преминаването на ядрото δ' от позиция S'_{31} в позиция S'_F. По същия начин при постъпване на ядрата θ'' и δ'' в общата позиция за преходите Z_2 и Z''_3 се активират предикатите $W_{S''31,S22}$ и $W_{S''31,S''F}$ за ядрата θ'' и δ''. Кратко описание с условията на всички предикати е представено в табл. 3.7.

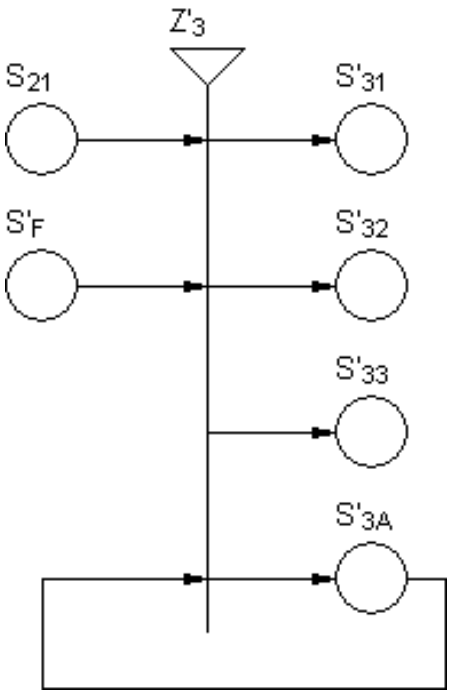
Предикат	Описание
$W_{S11,S21}$	Ако името на ядрото е θ' в позиция S_{11}
$W_{S11,S22}$	Ако името на ядрото е θ'' в позиция S_{11}
$W_{SF,S'F}$	Ако името на ядрото е δ' в позиция S_{SF}
$W_{SF,S''F}$	Ако името на ядрото е δ'' в позиция S_{SF}
$W_{S'31,S21}$	Ако името на ядрото е θ' в позиция S'_{31}
$W_{S'31,S'F}$	Ако името на ядрото е δ' в позиция S'_{31}
$W_{S''31,S22}$	Ако името на ядрото е θ'' в позиция S''_{31}
$W_{S''31,S''F}$	Ако името на ядрото е δ'' в позиция S''_{31}

Табл. 3.7. Описание на предикатите в R_2

С така представените характеристични функции и предикати за прехода Z_2 се извършва дублирането на информацията за невронната мрежа чрез разделяне на постъпващите ядра със съответните характеристични функции. Разделените ядра биват насочвани към преходите Z'_3 и Z''_3 чрез предикатите на прехода. По този начин задачата на прехода Z_2 е изпълнена.

3.3.3 Описание на преход Z'_3

Преходът Z'_3 служи за обучение на първата невронна мрежа от конкурентно обучаваните невронни мрежи. Условието на прехода е $\vee(\wedge(S_{21}, S'_F), S'_{3A})$, което изисква във входни позиции S_{21} и S'_F да са постъпили необходимите ядра за активиране на прехода. В позиция S_{21} постъпва ядрато θ‘. То носи структурата на невронната мрежа, която ще бъде използвана от характеристичните функции за обучение на невронната мрежа. В позицията S'_F постъпва ядро δ‘ при активирането на прехода Z'_3. Ядрото δ‘ носи преходните функции на трите слоя на невронната мрежа. Чрез информацията от това ядро характеристичните функции за изчислението на правия и обратния ход извършват изчисленията. В табл. 3.8 са представени ядрата, които постъпват на входните позиции на прехода.



Фиг. 3.13. Графично представяне на прехода Z′3

$Z'_3 = <\{S_{21}, S'_F, S'_{3A}\}, \{S'_{31}, S'_{32}, S'_{33}, S'_{3A}\}, R'_3, \vee(\wedge(S_{21}, S'_F), S'_{3A}) >$

Преход *Z′3* има четири характеристични функции, които се използват за обучението на невронната мрежа. Първата характеристична функция, която се изпълнява, е характеристичната функция *CharBPFeedForward-Calculation*. Тя изчислява правия ход от обучението на невронната мрежа.

Ядро	Начални характеристики	Позиция на създаване
$\boldsymbol{\theta}$'	$x_0^\theta, x_1^\theta, x_2^\theta, x_3^\theta, x_4^\theta, x_5^\theta, x_6^\theta, x_7^\theta, x_8^\theta$	S_{21}
$\boldsymbol{\delta}$'	x_0^δ	S'_F
$\boldsymbol{\lambda'_1}$	$x_0^{\lambda 1}$	S'_{32}
$\boldsymbol{\lambda'_2}$	$x_0^{\lambda 2}$	S'_{33}

Табл. 3.8. Таблица на ядрата в преход Z'3

Необходимо предусловие за изпълнението на *CharBPFeedForwardCalculation* е постъпването на ядрата θ' и δ' в позиции S_{21} и S'_F *съответно*. Резултатът от работата на характеристичната функция е промяна на характеристиката x_5^θ. В нея се съхранява наредена *n*-торка с изходния резултат от обучението на невронната мрежа в правия ход. При завършване на обучението на невронната мрежа се активират други две характеристични функции - *CharLambdaPrimOneTokenGeneration* и *CharLambdaPrimTwoTokenGeneration*. Характеристичната функция *CharLambdaPrimOneTokenGeneration* се използва за генериране на ново ядро λ'_1 на позиция S'_{32} с характеристика $x_0^{\lambda 1}$. Характеристиката $x_0^{\lambda 1}$ съхранява *L*, което представлява минималния брой скрити неврони за обучената невронна мрежа. Характеристичната функция *CharLambdaPrimTwoTokenGeneration* генерира ново ядро λ'_2 на позиция S'_{33} с характеристика $x_0^{\lambda 2}$. Тази характеристика съхранява *M*, което е максималният брой неврони за обучената невронна мрежа. Последната

характеристична функция е *CharBPBackwardCalculation*. Условията, за да се изпълни функцията, са в позиция S'_{3A} да се намира ядрото θ‘. Резултатът от работата на тази характеристична функция е промяна на матрицата с тегловните коефициенти ($\boldsymbol{x_7^{\theta}}$) и вектора с отместванията ($\boldsymbol{x_8^{\theta}}$).

Преминаването на ядрата от входни към изходни позиции в прехода Z'_3 се определя от матрицата на предикатите R'_3, която е представена на фиг. 3.14. Повечето условия за преминаване от входните към изходните позиции са константни.

R'_3	S'_{31}	S'_{32}	S'_{33}	S'_{3A}
S_{21}	False	False	False	True
S'_F	False	False	False	True
S'_{3A}	$W'_{3A,31}$	$W'_{3A,32}$	$W'_{3A,33}$	True

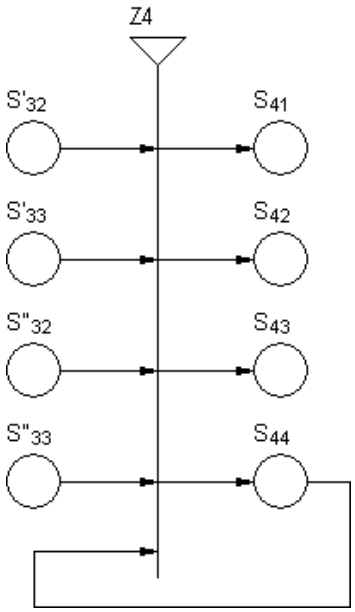
Фиг. 3.14. Матрица на предикатите R'_3

Важни за работата на прехода са предикатите, дефинирани за входната позиция S'_{3A}. В зависимост от типа на ядрото и условията, се избира в каква изходна позиция да премине ядрото. Предикатът $W'_{3A,31}$ разрешава преминаването на ядрата θ‘ и δ‘ от позиция S'_{3A} към позиция S'_{31} при условие, че текущата грешка от обучението на невронната мрежа E_1 е по-голяма от целевата грешка E_{max}. Преминаването в изходящата позиция S'_{31} позволява да се продължи обучението със следващата епоха, достигайки прехода Z_2. Предикатът $W'_{3A,32}$ разрешава преминаването на ядрата θ‘ и δ‘ към изходната позиция за прехода S'_{32}. Преминаването на ядрата към прехода Z_4 е продиктувано от факта, че обучението на невронната мрежа е постигнало желаните резултати. Последен от матрицата на преходите R'_3 е предикатът $W'_{3A,33}$. $W'_{3A,33}$ позволява преминаване на ядрата θ‘ и δ‘ в позицията S'_{33} на прехода Z_4, когато не е постигната целевата грешка E_{max} за обучението на невронната мрежа и е достигнат максималният брой епохи m за обучението на невронната мрежа. Характеристиката $x_0^{\lambda 3}$ на ядрото δ‘ съхранява стойността на E_{max}, а $x_0^{\lambda 4}$ съхранява стойността на m. Параметрите на невронната мрежа, текущата средноквадратична грешка на обучение E_1 и броят епохи на обучение, са новогенерирани характеристики x_9^{θ} и x_{10}^{θ} на ядрото θ‘ от характеристичните функции на обучение *CharBPFeedForwardCalculation* и *CharBPBackward-Calculation*. С представянето на предиката $W'_{3A,33}$ описанието на прехода Z'_3 е завършено. Функцията на прехода Z'_3 може да се формулира накратко като обучение на невронна мрежа с обратно разпространение на грешката.

3.3.5 Описание на преход Z_4

Z_4=<{S'_{32}, S'_{33}, S''_{32}, S''_{33}, S_{44}}, {S_{41}, S_{42}, S_{43}, S_{44}}, R_4, $\vee(S'_{32}, S'_{33}, S''_{32}, S''_{33}, S_{44})$ >

Преходът Z_4 служи за обобщаване на резултатите от обучението на невронните мрежи и взимане на решение за следващата стъпка от алгоритъма *златно сечение*. Необходимото и достатъчно условие за активиране на прехода Z_4 е $\vee(S'_{32}, S'_{33}, S''_{32}, S''_{33}, S_{44})$. Във входната позиция S'_{32} могат да постъпят ядра θ', δ' и λ'_1 от прехода Z'_3.



Фиг. 3.17. Графично представяне на прехода Z_4

Постъпването на ядра в тази позиция означава, че невронната мрежа в прехода Z'_3 е постигнала грешка по-малка от целевата средноквадратична грешка E_{max}. При постъпване на ядрата θ', δ' и λ'_2 в позиция S'_{33} се счита, че е прекратено обучението на невронната мрежа в прехода Z'_3 поради достигане на максималния брой епохи за обучение и средноквадратична грешка, превишаваща грешката E_{max}. Постъпването на ядрата θ'', δ'' и λ''_1 на входната позиция S''_{32} на прехода Z''_3 при активиране на предиката $W''_{3A,32}$ ($E_2 < E_{max}$) съвпада с поведението на прехода Z'_3 в позиция S'_{32}. Постъпване на ядрата θ'', δ'' и λ''_2 във входната позиция S''_{33} е възможно при по-голяма средноквадратична грешка от желаната целева грешка E_{max} в прехода Z''_3 при достигане на максималния позволен брой епохи на обучение. Допълнителна структурирана информация за ядрата е представена в табл. 3.16 по-долу. В позиция S_{44} постъпват всички ядра, пристигнали от позиции S'_{32}, S'_{33}, S''_{32} и S''_{33}. От натрупаната информация от ядрата в тази позиция се взема решение дали да се прекрати или да се продължи обучението на невронните мрежи в нов интервал от алгоритъма на *златното сечение*. При постъпване на ядрата от тип λ се активира характеристичната функция *CharGoldenSectionCalculation*.

Ядро	Начални характеристики	Позиция на създаване/постъпване
θ'	$x_0^\theta, x_1^\theta, x_2^\theta, x_3^\theta, x_4^\theta, x_5^\theta, x_6^\theta, x_7^\theta, x_8^\theta$	S'_{32} / S'_{33}
δ'	x_0^δ	S'_{32} / S'_{33}
λ'_1	$x_0^{\lambda 1}$	S'_{32}
λ'_2	$x_0^{\lambda 2}$	S'_{33}
θ''	$x_0^\theta, x_1^\theta, x_2^\theta, x_3^\theta, x_4^\theta, x_5^\theta, x_6^\theta, x_7^\theta, x_8^\theta$	S''_{32} / S''_{33}
δ''	x_0^δ	S''_{32} / S''_{33}
λ''_1	$x_0^{\lambda 1}$	S''_{32}
λ''_2	$x_0^{\lambda 2}$	S''_{33}
$\imath$	$x_0^\imath$	S_{44}

Табл. 3.16. Таблица на ядрата в преход Z_4

На базата на характеристиките $x_0^{\lambda 1}$ и $x_0^{\lambda 2}$ се взима решение за преминаване към стъпка 2 от алгоритъма на *златното сечение* (вж. точка 3.1). Резултатът от характеристичната функция *CharGoldenSectionCalculation* е генерирането на ядро *ı* и сливане на ядрата *θ′*, *δ′*, *θ″* и *δ″* отново до ядра *θ* и *δ* със съответните параметри. Новосъздаденото ядро *ı* притежава една характеристика $x_0^{\imath}$. Тази характеристика съдържа минималната и максималната стойност *L* и *M* на броя скрити неврони от интервала на алгоритъма *златно сечение*. Слетите ядра *θ* и *δ* запазват основните параметри, за да се инициира процесът на обучение в прехода Z_1 отначало. Преминаването на ядрата между входните и изходните позиции на прехода Z_4 се определя от матрицата на преходите R_4, която е представена на фиг. 3.18 по-долу.

R_4	S_{41}	S_{42}	S_{43}	S_{44}
S'_{32}	False	False	False	True
S'_{33}	False	False	False	True
S''_{32}	False	False	False	False
S''_{33}	False	False	False	True
S_{44}	$W_{44,41}$	$W_{44,42}$	$W_{44,43}$	True

Фиг. 3.18. Матрица на предикатите R_4

Както във всяка матрица, така и в тази има константно зададени разрешения и забрани за преминаване между позициите. Матрицата на преходите има три важни предиката $W_{44,41}$, $W_{44,42}$ и $W_{44,43}$, от които зависи вземането на решение за поведението на целия обобщеномрежов модел. Първият предикат $W_{44,41}$ разрешава преминаването на ядрата от позиция S_{44} в позиция S_{41}. Условието за преминаването на ядрата *θ′* и *θ* е стойностите на средноквадратичните грешки от обучението на невронните мрежи $E_1(x_9^{\theta})$ и $E_2(x_9^{\theta})$ да бъдат по-малки от целевата средноквадратична грешка E_{max}. Преминаването в позиция S_{41} се приема като край на работата на обобщеномрежовия модел с постигане на желаната целева средноквадратична грешка E_{max}. Преходът $W_{44,42}$ разрешава преминаването на ядрата от позиция S_{44} към позиция S_{42}, когато конкурентно обучаваните невронни мрежи не са постигнали желаната стойност на E_{max} в процеса на обучение. Преминаването на ядрата в тази позиция също се приема за край на обучението. Последният предикат в матрицата на преходите R_4 е $W_{44,43}$. Предикатът $W_{44,43}$ е *истина* в случаите, в които първата или втората конкурентно обучавани невронни мрежи са постигнали целевата грешка E_{max}. Постъпването на ядрата в тази позиция се приема за стъпка 2 от алгоритъма на *златното сечение.*Резултатът е преминаване на ядрата *ı*, *θ* и *δ* в позиция S_{43}, която се явява входна за прехода Z_1 и начало на обучение на двете невронни мрежи с новия интервал на *златното сечение*.

В табл. 3.18 по-долу са описани формално предикатите от матрицата на преходите R_4. Преходът Z_4 може да се класифицира като контролиращо звено за изпълнение на алгоритъма *златно сечение*.

Предикат	Описание
$W_{44,41}$	$(E_1<E_{max})$ & $(E_2<E_{max})$
$W_{44,42}$	$(E_1>E_{max})$ & $(n_1>m)$ & $(E_2>E_{max})$ & $(n_2>m)$
$W_{44,43}$	$((E_1<E_{max})$ & $(E_2>E_{max})$ & $(n_2>m))$ | $((E_2 < E_{max})$ & $(E_1 > E_{max})$ & $(n_1 > m))$

Табл. 3.18. Описание на предикатите в R_4

3.4 Постигнати резултати

Оптимизиране на обучението чрез употреба на алгоритъма златно сечение

Алгоритъмът на *златно сечение* предполага оптимизация по един параметър. В разгледания по-горе модел оптимизацията е направена по броя на скритите неврони за невронната мрежа с право разпространение на сигнала. Причината да се ограничи обобщеномрежовият модел да симулира работата на две конкурентно обучавани невронни мрежи е породена от изискванията на алгоритъма да използва долна и горна граница на интервала за оптимално търсене на най-добре обучена невронна мрежа. Предимствата на този алгоритъм са известни, за повече подробности виж [41].

Ефективно използване на ресурсите на съвременните процесори

Съвременните процесори използват повече от едно ядро в един корпус. Причината за това е невъзможността да се увеличава честотата на процесорите с досегашните темпове технологично развитие [158, 159]. Наличието на две и повече ядра в корпуса на всеки процесор води до нов начин на изпълнение на всеки софтуер, както и изисквания към използваните алгоритми за машинно обучение да използват максимално ефективно наличните ядра на всеки процесор.

Ускорение процеса на обучение на невронната мрежа

Ускорението на процеса на обучение на невронната мрежа с право разпространение на сигнала се постига чрез конкурентно обучение на две невронни мрежи. Причината да се избягва паралелно обучение на невронна мрежа с право разпространение на сигнала е невъзможността да се изпълняват паралелно стъпки от алгоритъма за обучение, които водят до сериозна скалируемост на изчислението. Обобщеномрежовият модел от тази глава е публикуван в [21].

Глава 4. Обобщеномрежов модел за паралелна оптимизация на скрити неврони в невронна мрежа с радиални базисни функции

Цел на настоящата глава от дисертационния труд е да се създаде обобщеномрежов модел за паралелна оптимизация на невронни мрежи с

радиални базисни функции. Така поставената цел изпълнява задачи 1 и 4 от задачите на дисертационния труд.

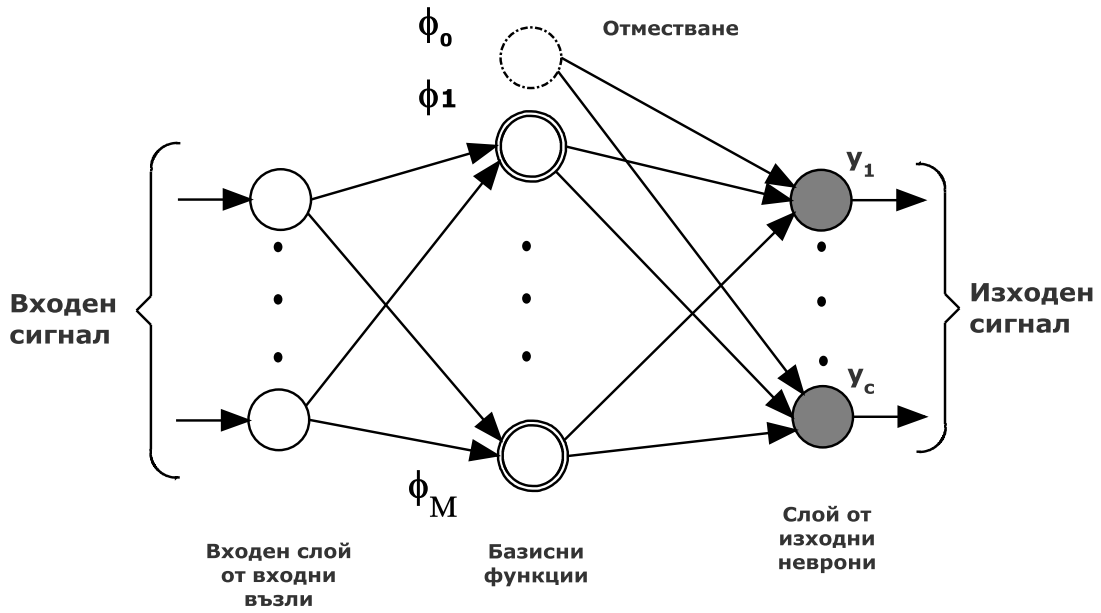
В последните години се провеждат дискусии относно наличието на голям брой ядра в процесорите, използвани в ежедневието [158]. При условие че съвременните алгоритми са последователни, в повечето случаи те не могат да натоварят максимално съществуващите хардуерни възможности за паралелна обработка. Примери за достъпни средства за паралелна обработка са компютърни системи с многоядрени процесори [125], клъстери от сървъри [38], изчислителни системи с няколко мощни видеокарти [58] и не на последно място изчислителни технологии от тип „облак" [72]. Тези средства позволяват да бъдат потърсени нови подходи в обучението на невронни мрежи и оптимизация на съществуващи алгоритми с цел максимално използване на наличните ресурси за паралелно изчисление.

В направения в т. 1.3 на настоящата дисертация литературен обзор на публикуваните вече невронни мрежи и подходи за машинно обучение (включително и IEEE архивите [69, 70]) липсват материали за паралелно обучение на невронни мрежи с радиални базисни функции. Представеният по-долу модел за паралелна обработка цели да намали времето за обучение на невронната мрежа с радиални базисни функции чрез едновременно обучение на няколко невронни мрежи. Възможно е процесът на обучение да бъде преустановен ако дадена невронна мрежа е достигнала желан резултат. Моделът за паралелна обработка е формално описан в термините на обобщени мрежи [15, 19, 2, 22]. Това позволява реална паралелна симулация на представения модел с формално математическо описание. Паралелната симулация на обобщеномрежов модел използва обща времева скала, чрез която може да бъде сравнена скоростта на обучение на конкретна невронна мрежа при паралелно и последователно изпълнение.

4.1 Невронна мрежа с радиални базисни функции

Разгледаният по-горе невронен мрежов модел се базира на неврони, които изчисляват нелинейна функция на скаларно произведение между входните и тегловните стойности на невронната мрежа. Тук ще бъде разгледан друг клас невронен мрежов модел, в който активацията на един скрит неврон се определя от разстоянието между входния вектор и вектора на прототипа за обучение.

Интересно и важно свойство на мрежите с радиални базисни функции е, че те формират обединяващо звено между определен брой коренно различни концепции, както ще бъде показано по-нататък. Ще бъде обоснована употребата на радиални базисни функции от гледна точка на апроксимацията на функции, регулярности, шумова интерполация, плътностна оценка, оптималната класификационна теория и потенциални функции.



Фиг. 4.2. Архитектура на невронна мрежа с радиални базисни функции

Едно следствие от тази обединяваща гледна точка, е че това полага основата за обучението на мрежи с радиални базисни функции, които могат да бъдат изключително бързи в сравнение с мрежи като многослойния перцептрон. Това следва от интерпретацията, която може да бъде дадена от вътрешното представяне, формирано от скритите неврони и водещо до двуфазен алгоритъм за обучение. В първата фаза параметрите, управляващи базови функции (съответстващи на скрити неврони), се определят чрез употребата на относително бързи методи за обучение без наблюдение. Втората фаза на обучение включва определянето на теглата в последния слой, което изисква решението на линеен проблем, който също е бърз.

4.1.3.1 Алгоритъм за обучение на невронни мрежи с радиални базисни функции

Тук ще бъде разгледан алгоритъм за обучение на невронна мрежа с радиални базисни функции с някои подобрения [29]. Подобренията на алгоритъма са свързани с употребата на допълнителен адаптивен алгоритъм наречен *RPROP*, създаден от Ридмилер и Браун [127] с цел по-бързо обучение в практически задачи. В тази версия на алгоритъма [29] се предоставя възможност за промяна на топологията на невронната мрежа чрез динамично увеличаване и намаляване на броя гаусови неврони в скрития слой. Топологичната промяна на мрежата позволява достигане на по-точна интерполация на резултатите.

Алгоритъмът за обучение на мрежата преминава през няколко основни етапа, които са типични за обучението на невронните мрежи. По-долу ще бъдат разгледани последователно стъпките за обучение, както за една епоха, така и за целия процес на обучение. За целта на обучението се изпълняват следните стъпки:

1. **Инициализация.** В тази стъпка се инициализират основните параметри на невронната мрежа като: брой входове, скрити неврони, изходни неврони, целева грешка на обучението, брой епохи, $\mu_{dist_{del}}$- минимално разстояние на математическото очакване между два гаусови неврона и w_{del}, σ_{del}, μ_{del} - прагови стойности за изтриване на гаусови неврони. За всеки гаусов неврон се инициализират σ_j, $\Delta\sigma_j$, μ_j и $\Delta\mu_j$, а за изходящите неврони w_{kj}- тегловните коефициенти, Δw_{kj}– изменението на тегловните коефициенти, θ – прагова стойност и нейното изменение $\Delta\theta$.
2. **Обучение на невронната мрежа.**
   - ***2.1. Обучение на епоха***
     - **2.1.1.** ***Запазване на натрупаната информация от предишната епоха.*** В тази стъпка се съхраняват изчисленията от предишната епоха свързани с ***E*** като $\sum_{k=0}^{n}\sum_{j=0}^{g}\frac{\partial E(t)}{\partial w_{kj}}$, $\sum_{k=0}^{n}\frac{\partial E(t)}{\partial \theta_k}$, $\sum_{j=0}^{g}\sum_{i=0}^{d}\frac{\partial E(t)}{\partial \mu_{ji}}$ и $\sum_{j=0}^{g}\frac{\partial E(t)}{\partial \sigma_j}$, където *t* отразява конкретен набор за обучение, индекс *i* е номерът на входния възел (неврон), *j* е номерът на скрития неврон (гаусова базисна функция), а *k* е номерът на изходния неврон.
     - **2.1.2.** ***Изчисляване на входното въздействие върху изхода за даден набор на обучение.*** В тази стъпка първоначално се изчислява резултатът на всяка гаусова базисна функция $\phi_j(x)$, като се използват входните стойности от входните възли (вж. 4.15).

$$\phi_j(x) = exp\left(-\frac{\|x-\mu_j\|^2}{2\sigma_j^2}\right) \qquad (4.15)$$

       След това се изчислява изходният резултат $y_k(x)$ за всеки изход на невронната мрежа, като се използва $\phi_j(x)$ и се умножи по тегловен коефициент w_{kj}, където

$$y_k(x) = \sum_{j=0}^{M} w_{kj}\phi_j(x). \qquad (4.17)$$

       От получения резултат $y_k(x)$ се изважда праговата стойност θ_k за дадения изходен неврон. Така изчисленият краен резултат съответства на стойностите, получени на изхода на невронната мрежа.
     - **2.1.3.** ***Изчисляване на E на изхода от приложеното входно въздействие.*** Изчисляването на ***E*** представлява изчисляване на квадратична оценъчна функция

$$\mathrm{E} = \frac{1}{2}\sum_n\sum_k\{y_k(\boldsymbol{x}^n) - t_k^n\}^2, \qquad (4.19)$$

       където t_k^n е целевата стойност за изходния неврон *k*, когато мрежата е представена с входен вектор $\boldsymbol{x}^n$. В тази стъпка се проверява дали изчислената грешка за даден обучителен набор е най-голямата от досега изчислените. Ако това е вярно, се записва стойността на текущата грешка като най-голямата натрупана такава за даден набор, както и входните и изходните стойности на

наборите за обучение. Целта на тази проверка е да се използва съхранената информация в следващите стъпки за обучение на невронната мрежа.

2.1.4. ***Изчисляване на промяната на E спрямо параметрите на мрежата*** w_{kj}, $\Delta\theta_k$, $\Delta\mu_{ji}$ ***и*** $\Delta\sigma_j$. В тази стъпка се изчислява влиянието на посочените по-горе параметри, които взимат пряко участие в процеса на обучение и върху изменението на ***E***. Целта на тази стъпка е да изчисли $\sum_{k=0}^{n}\sum_{j=0}^{g}\frac{\partial E(t)}{\partial w_{kj}}$, $\sum_{k=0}^{n}\frac{\partial E(t)}{\partial \theta_k}$, $\sum_{j=0}^{g}\sum_{i=0}^{d}\frac{\partial E(t)}{\partial \mu_{ji}}$ и $\sum_{j=0}^{g}\frac{\partial E(t)}{\partial \sigma_j}$, които се използват от *RPROP* модификацията на алгоритъма за обучение [127].

2.1.5. ***Проверка за броя на наборите, приложени върху невронната мрежа.*** В този етап от обучението на невронната мрежа се проверява дали има други набори за обучение. Ако има такива, се избира нов набор за обучение и се преминава към стъпка 2.1.1. В случай че няма нови набори за обучение, се преминава към следващата стъпка от алгоритъма.

2.1.6. ***Изчисляване на грешката от текущата епоха.*** След като са изчислени грешките за всяка една епоха в т. 2.1.3, следва да се сумират към общата грешка и да се раздели на броя набори за обучение. Ако получената обща грешка е по-малка от най-малката такава до момента, то параметрите на текущата невронна мрежа се записват и се отбелязва епохата, в която е постигнат този резултат.

2.1.7. ***Промяна на параметрите на изходните неврони.*** В тази стъпка от обучението на невронната мрежа се коригират праговите стойности и тегловните коефициенти на всеки изходен неврон. Последователността от стъпки за корекция на параметрите на всеки изходен неврон е следната:

- Изчисляване на коригиращ фактор c_w на Δw_{kj} на тегловен коефициент w_{kj}. Ако произведението $\frac{\partial E}{\partial w_{kj}} * \frac{\partial E_{old}}{\partial w_{kj}}$ е положително, то на c_w се присвоява стойност η^+, в случай че е отрицателно се присвоява η^-, а при нулев резултат се присвоява единица.
- Изчисленият коригиращ фактор c_w се умножава с текущото изменение на тегловния коефициент Δw_{kj} и се получава новото изменение $\Delta w_{kj_{new}} = \Delta w_{kj} * c_w$.
- $\Delta w_{kj_{new}}$ се използва за корекция на тегловния коефициент w_{kj} чрез следното уравнение: $w_{kj_{new}} = w_{kj} - (\Delta w_{kj_{new}} * \frac{\partial E}{\partial w_{kj}})$.
- Изчисляване на коригиращ фактор c_θ на $\Delta\theta_k$ на праговата стойност θ_k. Ако произведението $\frac{\partial E}{\partial \theta_k} * \frac{\partial E_{old}}{\partial \theta_k}$ е положително, то на c_θ се присвоява стойност η^+, в случай че е отрицателно, се присвоява η^-, а при нулев резултат се присвоява единица.

- Новата стойност на $\Delta\theta_{k_{new}}$, се получава като се изчисли следното уравнение: $\Delta\theta_{k_{new}} = \Delta\theta_k * c_\theta$.
- $\Delta\theta_{k_{new}}$ се употребява, за да коригира праговата стойност θ_k на даден изходен неврон чрез следното уравнение: $\theta_{k_{new}} = \theta_k - (\Delta\theta_{k_{new}} * \frac{\partial E}{\partial \theta_k})$.

2.1.8. ***Промяна на параметрите на гаусовите неврони.*** Тази стъпка от обучението на алгоритъма е подобна на горната. Тук се коригират математическото очакване μ_{ji} на всеки гаусов неврон и неговото стандартно отклонение σ_j. Последователността от стъпки за корекция на параметрите на всеки гаусов неврон е следната:

- Изчисляване на коригиращ фактор c_μ на $\Delta\mu_{ji}$ на математическото очакване μ_{ji}. Ако произведението $\frac{\partial E}{\partial \mu_{ji}} * \frac{\partial E_{old}}{\partial \mu_{ji}}$ е положително, то на c_μ се присвоява стойност η^+, в случай че е отрицателно, се присвоява η^-, а при нулев резултат се присвоява единица.
- Изчисленият коригиращ фактор c_μ се умножава с текущото изменение на математическото очакване $\Delta\mu_{ji}$ и се получава новото изменение $\Delta\mu_{ji_{new}} = \Delta\mu_{ji} * c_\mu$.
- $\Delta\mu_{ji_{new}}$ се използва за корекция на математическото очакване μ_{ji} чрез следното уравнение: $\mu_{ji_{new}} = \mu_{ji} - (\Delta\mu_{ji_{new}} * \frac{\partial E}{\partial \mu_{ji}})$.
- Изчисляване на коригиращ фактор c_σ на $\Delta\sigma_j$ на стандартно отклонение σ_j. Ако произведението $\frac{\partial E}{\partial \sigma_j} * \frac{\partial E_{old}}{\partial \sigma_j}$ е положително, то на c_σ се присвоява стойност η^+, в случай че е отрицателно се присвоява η^-, а при нулев резултат се присвоява единица.
- Новата стойност на $\Delta\sigma_{j_{new}}$ се получава като се изчисли следното уравнение: $\Delta\sigma_{j_{new}} = \Delta\sigma_j * c_\sigma$.
- $\Delta\sigma_{j_{new}}$ се употребява, за да коригира стандартното отклонение σ_j за даден скрит неврон чрез следното уравнение: $\sigma_{j_{new}} = \sigma_j - (\Delta\sigma_{j_{new}} * \frac{\partial E}{\partial \sigma_j})$.
- Ако стандартното отклонение $\sigma_j \geq 0$, то σ_j и μ_{ji} на неврона се инициализират с нови стойности.
- Ако $\mu_{ji} > \mu_{del}$, то гаусовият неврон се изтрива.
- Ако $w_{kj} > w_{del}$, то гаусовият неврон се изтрива.
- Ако $\sigma_j < \sigma_{del}$, то гаусовият неврон се изтрива.
- Ако разстоянието между два скрити неврона $\mu_{dist}(\mu_{j1i1}, \mu_{j2i2}) > \mu_{dist_{del}}$, то скритият неврон, съдържащ μ_{j1i1}, се изтрива.

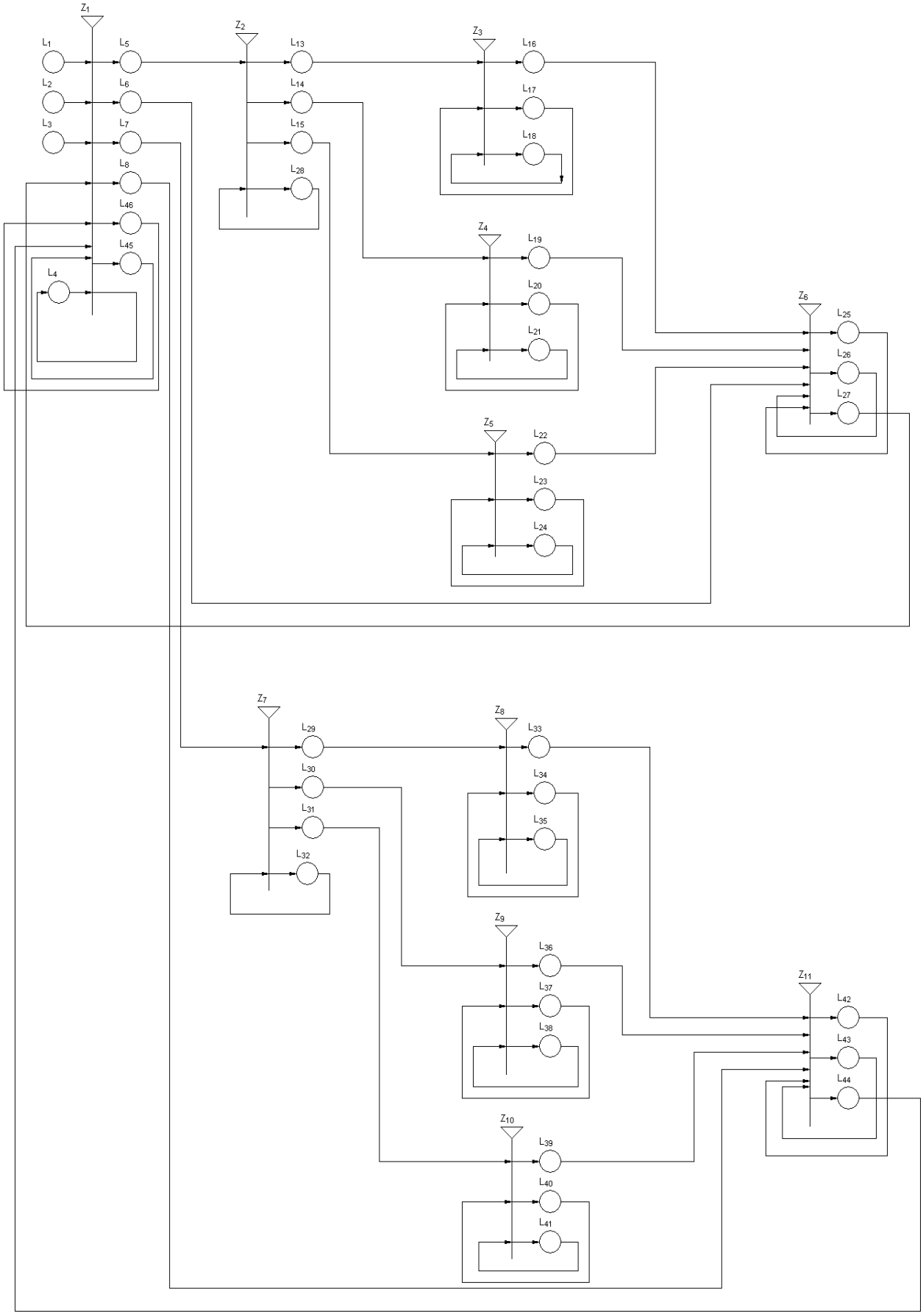
2.2. Проверка дали е достигнат краят на обучението

В тази стъпка се прави проверка дали е достигнат броят епохи, които са предвидени за обучение на невронната мрежа. Ако е

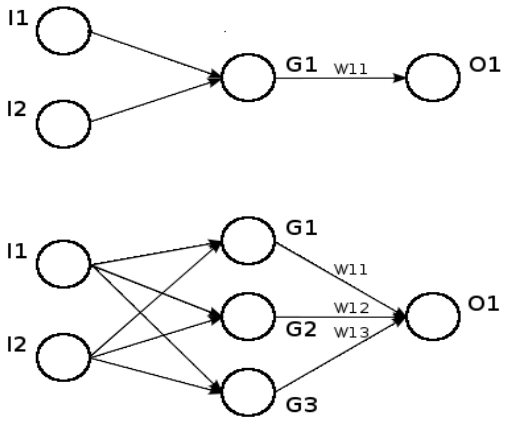
достигнат необходимият брой епохи, се прекратява процесът на обучение. Във всички останали случаи се преминава към точка 2.1. от алгоритъма.

4.2 Описание на модела

Обобщеномрежовият модел за обучение на невронни мрежи с радиални базисни функции е показан на фиг. 4.4. по-долу и може да бъде представен посредством три типа дейности. Първата дейност, която се изпълнява, е обработката на новопостъпилите входни ядра. Те съдържат пълната информация за всяка невронна мрежа, която трябва да бъде симулирана. Ядрата, които пристигат на входните позиции на преходите, са входни и изходни стойности, обучителни параметри, оценъчни параметри, ядра, носещи информация за гаусовите неврони, както и ядра които ще съхраняват най-добре обучените невронни мрежи. Резултатът от обработката на тези ядра е динамично генериране на нови ядра, които заедно с останалите преходи са необходими за представяне на всяка една от невронните мрежи с подходящите параметри. След генерирането на новите ядра за невронните мрежи е необходимо да се симулира паралелният процес за обучение на невронни мрежи. Втората дейност, която се изпълнява, е конкурентното обучение на невронните мрежи. Третата дейност се състои в проследяването и съхраняването на параметрите на най-добре обучената мрежа. Поради факта, че обобщените мрежи могат да описват паралелни процеси, те са добър инструмент за създаване на модел за симулация на повече от една невронна мрежа. Основите на невронните мрежи с радиални базисни функции са разгледани в 4.1. Тук ще бъдат описани конкретни невронни мрежи, които са симулирани в обобщеномрежовия модел, а именно две паралелно обучавани невронни мрежи за по-голяма яснота на симулационния процес и обратно проследяване на резултатите от обучението. Двете симулирани невронни мрежи се различават по броя на гаусовите неврони в скрития слой и по допълнителните инициализационни параметри. Състоят се от два входа и един изход, както е показано на фиг. 4.5 по-долу. Причината за избор на сравнително малки мрежи за конкурентно обучение е, че целта на симулацията е да се потвърди ползата от паралелната симулация на невронни мрежи и да се проследи сравнително лесно процесът на обучение. За отбелязване е, че изграждането на един модел на невронна мрежа е динамично, което определя голямата сложност на обобщеномрежовия модел. Множествата от данни за обучение се състоят от данни, които не могат да бъдат представени чрез непрекъсната функция (фиг. 4.6). Тези данни позволяват също така и да бъде наблюдавана конкурентност между обучаемите невронни мрежи по време на процеса на обучение.



Фиг. 4.4. Обобщеномрежов модел на невронна мрежа с радиални базисни функции



Фиг. 4.5. Невронна мрежа с радиални базисни функции

Номер на набора	Входен набор		Изходен набор
	*I*1	*I*2	*O*1
1	0.1	0.1	0.6
2	0.2	0.2	0.7
3	0.3	0.3	0.8

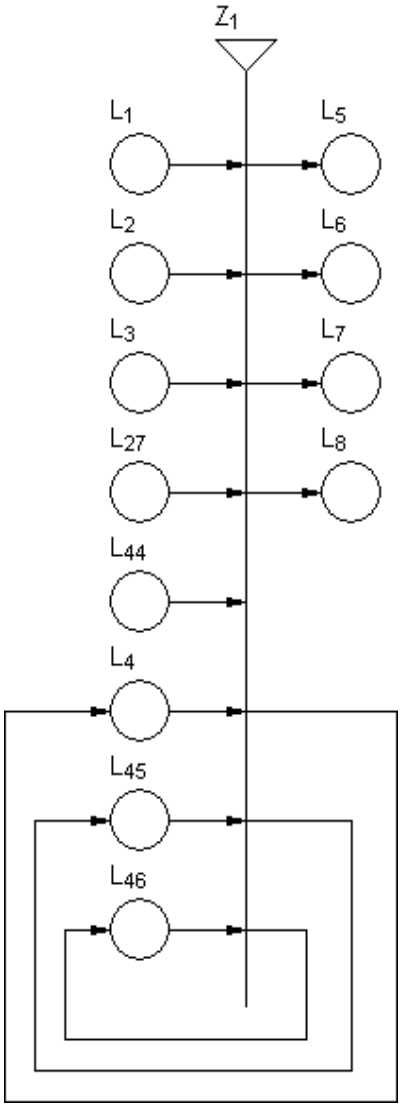
Фиг. 4.6. Набори за обучение

4.2.1 Описание на преход Z_1

$$Z_1 = <\{L_1, L_2, L_3, L_4, L_{27}, L_{44}, L_{45}, L_{46}\}, \{L_4, L_5, L_6, L_7, L_8, L_{45}, L_{46}\}, R_1,$$
$$\vee (\wedge (L_1, L_3, L_4), L_{27}, L_{44}, L_{45}, L_{46})>$$

Преходът Z_1 организира работата на алгоритъма преди стартиране на конкурентния процес на обучение на невронните мрежи с радиални базисни функции и основната част от процеса на обучението им. Първоначалните стъпки преди да започне процесът на обучение включват динамично генериране на подходящите ядра за обучение в двете невронни мрежи. Това са ядрата *LearningInputPatternNetOnePrim*, *LearningInputPattern-NetTwoPrim*, *LearningInputPatternNetTwoSecond*, *LearningInputPatternNet-TwoTerz*, *LearningOutputPatternNetOne* и *LearningOutputPatternNetTwo*. Тяхното генериране се определя от постъпването на следните ядра: *Inputs*, *Outputs*, *NetworkParametersOne*, *NetworkParametersTwo*, *ErrorParametersNet-One*, *ErrorParametersNetTwo*, *LearningParametersNetOne*, *LearningParameters-NetTwo*. Информация за ядрата е дадена в табл. 4.1. В прехода се изчисляват корекциите на параметрите за обучение след всяка една итерация или епоха в процеса на обучение. Освен това, с оптимизационна цел се натрупват и резултатите от обучението. За активирането на прехода Z_1 е необходимо на позициите L_1 да са постъпили ядрата *Inputs* и *Outputs*, в позиция L_3 да са постъпили ядрата *NetworkParametersOne* и NetworkParametersTwo, а в позиция L_4 да са постъпили ядрата *ErrorParametersNetOne*, *ErrorParametersNetTwo*, *Learning-ParametersNetOne* и LearningParametersNetTwo. При постъпване на ядра в някои от входните позиции се активират различни характеристични функции, които се

използват за обработката на характеристиките на ядрата. Характеристичните функции в прехода Z_1 могат да се разделят на два типа.



Фиг. 4.7. Графично представяне на прехода Z_1

Ядро	Начални характеристики	Позиция на създаване/постъпване
Inputs	*Inputs*, *NumberOfInputs*, *MaxPatterns*, *CurrentPattern*	L_1
Outputs	*Outputs*, *NumberOfOutputs*, *MaxPatterns*, *CurrentPattern*	L_1
Network-ParametersOne	*CurrentEpoch*, *DefaultSigma*, *Epoches*, *EtaMinus*, *EtaPlus*, *GaussNeurons*, *InputDimen-sion*, *OutputDimension*, *LearningPatterns*, *MuDelta*, *MuDistanceDelta*, *NumberOfInputs*, *NumberOfOutputs*, *MaxGauss-Neurons*, *MaxErrorPerPattern*, *SigmaDelta*, *WDelta*	L_3
Network-	*CurrentEpoch*, *DefaultSigma*, *Epoches*,	L_3

ParametersTwo	*EtaMinus*, *EtaPlus*, *GaussNeurons*, *InputDimen-sion*, *OutputDimension*, *LearningPatterns*, *MuDelta*, *MuDistanceDelta*, *Number-OfInputs*, *NumberOfOutputs*, *MaxGauss-Neurons*, *MaxErrorPerPattern*, *SigmaDelta*, *WDelta*	
LearningInput-PatternNetOne-Prim	*GaussNeuronId, Id, Type, Value, Inputs*	L_3
LearningInput-PatternNetTwo-Prim	*GaussNeuronId, Id, Type, Value, Inputs*	L_3
LearningInput-PatternNetTwo-Second	*GaussNeuronId, Id, Type, Value, Inputs*	L_3
LearningInput-PatternNetTwo-Terz	*GaussNeuronId, Id, Type, Value, Inputs*	L_3
LearningOutput-PatternNetOne	*Id*, *Type*, *Output*	L_3
LearningOutput-PatternNetTwo	*Id*, *Type*, *Output*	L_3
ErrorParameters-NetOne	*ErrorInsert*, *ErrorWindow*, *ErrorWindowSize*, *MaxErrorOutput*, *MaxErrorPattern*, *MaxErrorPerPattern*, *MinError*, *NumTrainedPatterns*, *WindowIndex*	L_4
ErrorParameters-NetTwo	*ErrorInsert*, *ErrorWindow*, *ErrorWindowSize*, *MaxErrorOutput*, *MaxErrorPattern*, *MaxErrorPerPattern*, *MinError*, *NumTrainedPatterns*, *WindowIndex*	L_4
LearningPara-metersNetOne	*MaxErrorOutput*, *SumOfdEdMu*, *SumOfdEdSigma*, *SumOfdEdThr*, *SumOfdEdW*, *OldSumOfdEdMu*, *OldSumOfdEdSigma*, *OldSumOfdEdThr*, *OldSumOfdEdW*	
LearningPara-metersNetTwo	*MaxErrorOutput*, *SumOfdEdMu*, *SumOfdEdSigma*, *SumOfdEdThr*, *SumOfdEdW*, *OldSumOfdEdMu*, *OldSumOfdEdSigma*, *OldSumOfdEdThr*, *OldSumOfdEdW*	

Табл. 4.1. Таблица на ядрата в преход Z_1

Първият тип са такива, които се извикват еднократно за генериране на подходящите ядра с информация за структурите на двете невронни мрежи. Такъв тип характеристична функция е *INPUTCharL3L5*. Нетипично при нея е моментът й на активиране спрямо този на всички останали характеристични функции в ОМ модел. Повечето характеристични функции се активират при постъпване на ядро в изходната позиция. В случая,

характеристичната функция *INPUTCharL3L5* се активира при постъпване на ядро от тип *NetworkParametersOne* или *Network-ParametersTwo* във входната позиция L_5. След активирането на функцията се изпълняват две стъпки за инициализация на двете невронни мрежи. Първата стъпка е:

- Да се инициализира първоначалният брой гаусови неврони;
- Да се извлече $\Delta\sigma$ за всеки гаусов неврон;
- Да се извлече $\sigma_{default}$ за всеки гаусов неврон;
- Да се определи максималният брой набори за обучение;
- Да се инициализира μ за всеки гаусов неврон;
- Да се инициализират тегловите коефициенти на гаусовите неврони.

Следващата стъпка е генериране на необходимите ядра от тип *LearningInputPattern* и *LearningOutputPattern* за двете невронни мрежи спрямо характеристиките на ядрата *NetworkParametersOne* и *NetworkParametersTwo*. Новогенерираните ядра служат за пренасяне на входните и желаните изходни въздействия. Ядрата от тип *LearningInput-Pattern* се използват също така, за да предадат въздействието от гаусовия към изходния неврон. В зависимост от подадените параметри в ядрата *NetworkParametersOne* и *NetworkParametersTwo,* динамично се генерират определен брой ядра от тип *LearningInputPattern* за двете невронни мрежи, които ще бъдат обучавани. По същия начин се генерират ядра от тип *LearningOutputPattern* за невронните мрежи, като тяхната цел е да пренесат информация за желания резултат до изходните неврони. В обобщената мрежа гаусовите неврони са симулирани чрез еднотипните преходи Z_3, Z_6, Z_7 и Z_8, принципът на работа на които ще бъде разгледан по-долу. Причината в ОМ модела да не се създават динамично гаусовите неврони, т.е. преходите, динамично е невъзможността на софтуера за симулация да генерира динамично преходи. Изходните неврони на двете невронни мрежи в ОМ модела са представени чрез преходите Z_4 и Z_9 и начинът им на работа ще бъде представен по-долу.

Вторият тип характеристични функции са онези, които се използват за пресмятане на резултатие при всяка една итерация или епоха в процеса на обучение за двете невронни мрежи. Първата характеристична функция, която е от този тип, е *CharL1L5*. Тя се активира всеки път при постъпване на ядро *LearningInputPatternNetOnePrim* в позиция L_5. Във функцията се изчислява номера на предстоящата итерация в процеса на обучение. Резултатът от пресмятането е промяна на характеристиката *CurrentPattern* в ядрото *Inputs*. Функцията *CharL1L6* е следващата функция, която се активира при постъпване на ядро *LearningOutputPatternNetOne*. Нейната основна задача е да извършва корекция на обучителните параметри, на база натрупани резултати след всяка епоха за първата невронна мрежа. Функцията се активира при постъпване на ядрото *LearningOutputPattern-NetOne* в изходната позиция L_6. При завършване на всяка епоха се изпълняват следните дейности:

- Съхраняване на резултатите на текущо обучената невронна мрежа при достигане на по-малка средноквадратична грешка от текущо постигнатата до момента;

- Коригиране на обучителните параметри на невронната мрежа, намиращи се в гаусовите неврони и изходните такива;
- Записване с оптимизационна цел на текущо постигнатите грешки $\sum_o \sum_g \frac{dE}{dw_{og}}$, $\sum_g \sum_i \frac{dE}{d\mu_{gi}}$ и $\sum_g \frac{dE}{d\sigma_g}$ за всички гаусови неврони;
- Изтриване на гаусов неврон при определени условия на обучителните параметри.

Дейността по съхраняването на резултатите на невронната мрежа се състои в записването на параметрите на невронната мрежа в ядрото *BestNeuralNetwork*. Най-голямата сложност при симулирането на този модел е при корекцията на обучителните параметри на невронната мрежа. Характеристичната функция *CharL1L6* коригира параметрите на гаусовите и изходните неврони. Първата стъпка преди извършване на промяна на параметрите на ядрата е да се довършат изчисленията върху параметрите на изходните неврони $\sum_{o=0}^{k} \sum_{g=0}^{m} \frac{dE}{dw_{og}}$ и $\sum_{o=0}^{k} \frac{dE}{d\theta_o}$, където индекс *o* представлява броя на изходните неврони в мрежата, а индексът *g* отразява броя на гаусовите неврони в мрежата. Диференциалното уравнение $\frac{dE}{dw_{og}}$ отразява изменението на средноквадратичната грешка *E* при обучението на невронната мрежа спрямо тегловните коефициенти на всеки изходен неврон и съответния гаусов неврон w_{og}. Второто диференциално уравнение $\frac{dE}{d\theta_o}$ отразява промяната на средноквадратичната грешка *E* при изменение на праговата стойност на изходния неврон θ_o. В този момент от обучението се усредняват стойностите на натрупаните грешки спрямо броя набори за обучение. Причината да се пресметнат тук крайните резултати на изходните неврони е липсата на информация в изходните неврони за край на епохата и броя на наборите за обучение. След пресмятането на крайните резултати на изходните неврони следва усредняване на грешката спрямо броя на наборите за обучение и на гаусовите неврони за параметри $\sum_{g=0}^{m} \sum_{i}^{l} \frac{dE}{d\mu_{gi}}$ и $\sum_{g=0}^{m} \frac{dE}{d\sigma_g}$, където индексът *m* показва броя на гаусовите неврони, а индексът *l* отразява броя на входните неврони за невронната мрежа. Диференциалното уравнение $\frac{dE}{d\mu_{gi}}$ представя изменението на средноквадратичната грешка *E* на невронната мрежа спрямо изменението на *μ* за всеки гаусов неврон и конретния входен неврон. Другото диференциално уравнение $\frac{dE}{d\sigma_g}$ представя изменението на средноквадратичната грешка *E* спрямо изменението на сигма стойността σ_g за конкретния гаусов неврон.

Следващата стъпка в процеса на обучение на невронната мрежа е корекция на параметрите на изходните неврони, използвайки *RPROP* оптимизация [127]. Първоначално се извличат стойностите на параметрите η^+ и η^- от ядрото *NetworkParametersOne*. Тези параметри се използват като стимулиращ или подтискащ ефект за основните параметри, от които се влияе средноквадратичната грешка. Следващата стъпка е да се извлекат масивите, които съхраняват параметъра $\frac{dE}{dw_{og}}$ от текущата епоха и $\frac{dE_{old}}{dW\boldsymbol{old}_{og}}$ от

предишната епоха от ядрото *LearningParametersNetOne*. За всеки изходен неврон и за конкретния гаусов неврон се пресмята произведението $\frac{dE}{dw_{og}} * \frac{dE_{old}}{dw_{og}^{old}}$. Ако произведението е положително, то изменението на тегловния коефициент $\frac{dE}{dw_{og}}$ за конкретния гаусов неврон ще бъде умножено с η^+. В случай че произведението е отрицателно, ще се извърши корекция с η^-. При резултат нула не се извършва корекция. След пресмятане на dw_{og} се прави допълнителна проверка дали dw_{og} надвишава стойността 1000. Ако това е истина, то на dw_{og} се присвоява стойността 1000. При стойност на dw_{og} по-малка от нула се присвоява нула. Тези допълнителни корекции се правят, за да не се намалят броят епохи на обучение, ако изменението на тегловните коефициенти е толкова голямо, че може да забави сходимостта на процеса на обучение. След като са извършени промените на dw_{og}, се прилага резултатът върху съответния тегловен коефициент w_{og}. Процесът на промяна на тегловните коефициенти се счита за завършен след пресмятане на всички коефициенти dw_{og} за всеки гаусов неврон и прилагането на dw_{og} върху w_{og}.

Следващата стъпка при корекцията на параметрите са характеристиките $d\theta$ и θ. Първата стъпка е да се извлекат масивите, които съхраняват параметъра $\frac{dE}{d\theta_o}$ от текущата епоха и $\frac{dE}{d\theta_{\boldsymbol{old}o}}$ от предишната епоха от ядрото *LearningParametersNetOne*. За всеки изходен неврон се пресмята $\frac{dE}{d\theta_o} * \frac{dE}{d\theta_{\boldsymbol{old}o}}$. Ако произведението е положително, то изменението на праговата стойност $\frac{dE}{d\theta_o}$ за дадения изходен неврон ще бъде умножено с η^+. В случай че произведението е отрицателно, ще се извърши корекция с η^-. Вече пресметната стойност на $\frac{dE}{d\theta_o}$ се използва, за да се пресметне θ на конкретния изходен неврон. С така направените корекции върху параметрите на изходните неврони dw_{og}, w_{og} и $d\theta$, θ корекцията за изходните неврони е завършена.

Следващата стъпка, която предстои, е промяната на параметрите на гаусовите неврони. Първоначално се извличат параметрите $\frac{dE}{d\mu_{gi}}$ и $\frac{dE}{d\mu\boldsymbol{old}_{gi}}$ от ядрото *LearningParametersNetOne*. За всеки гаусов неврон и за конкретния входен неврон се пресмята новата стойност на $d\mu_{gi}$, използвайки по-горе описания *RPROP* подход с η^+ и η^-. Новополученото изменение $d\mu_{gi}$ се използва, за да промени коефициента μ_{gi}. След тази стъпка всички коефициенти μ_{gi} са променени за следващата епоха на обучение.

Последната стъпка по отношение на корекциите на параметрите на гаусовите неврони са σ и $d\sigma$. Първоначално се извличат $\sum_{g=0}^{l}\sum_{i}^{k}\frac{dE_{\boldsymbol{old}}}{d\sigma_{\boldsymbol{old}}}$ и $\sum_{g=0}^{l}\sum_{i}^{k}\frac{dE}{d\sigma}$. Ако произведението $\sum_{g=0}^{l}\sum_{i}^{k}\frac{dE_{\boldsymbol{old}}}{d\sigma_{\boldsymbol{old}}} * \sum_{g=0}^{l}\sum_{i}^{k}\frac{dE}{d\sigma} > 0$, то $d\sigma = d\sigma * \eta^+$. Ако $\sum_{g=0}^{l}\sum_{i}^{k}\frac{dE_{\boldsymbol{old}}}{d\sigma_{\boldsymbol{old}}} * \sum_{g=0}^{l}\sum_{i}^{k}\frac{dE}{d\sigma} < 0$, то $d\sigma = d\sigma * \eta^-$. Така получената $d\sigma$ се използва за корекция на σ по следния начин: $\sigma_{new} = \sigma - \sigma * d\sigma$. Ако $\sigma_{new} < 0$, на текущия гаусов неврон се инициализират всички μ_{gi} с произволни стойности от интервала [0, 1]. Едновременно с генерирането на нови стойности на всички μ_{gi} се генерира и нова произволна стойност за σ_{gnew}, която е в

интервала [$d\sigma_g$, $\sigma_{default}$]. Параметърът $\sigma_{default}$ е предварително зададен преди началото на обучението като характеристика в ядрото *LearningParametersNetOne*. В случай че $d\mu_{gi} > \mu_{\Delta max}$, то гаусовият неврон се изтрива. Параметърът $\mu_{\Delta max}$ е предварително зададен и съхранен в ядрото *LearningParametersNetOne*. В зависимост от стойността на dw_{og}, се изтрива даденият гаусов неврон при следните условия:

- ако $dw_{og} = 1$;
- ако dw_{og} и $\sigma_{new} < d\sigma$.

След като са направени всички корекции в процеса на обучение, се съхраняват следните стойности за *RPROP* оптимизация:

- $\sum_{o=1}^{m}\sum_{g=1}^{l}\frac{dE}{dw_{og}}$ за всички изходни неврони;
- $\sum_{g=1}^{l}\sum_{i=1}^{k}\frac{dE}{d\mu_{gi}}$ за всички гаусови неврони;
- $\sum_{g=1}^{l}\frac{dE}{d\sigma_g}$ за всички гаусови неврони.

С тези последни стъпки завършва работата на характеристичната функция *CharL1L6*.

Следващата характеристична функция в прехода Z_1 е *CharL1L8*. Тя се активира при постъпване на ядро *LearningParametersNetTwo* в позиция L_8. Тази характеристична функция изпълнява същата последователност от стъпки за втората невронна мрежа, както характеристичната функция *CharL1L6* за първата невронна мрежа. Поради този факт няма да бъде описвана подробно нейната работа. Следващата функция, която се изпълнява в прехода Z_1, е *CharL1L45*. При постъпване на ядро в позиция L_{45} се активира характеристичната функция. Ако постъпилото ядро е от тип *LearningInputPattern* и неговото ID е *LearningInputPatternNetOnePrim*, се изчисляват параметри $\frac{dE}{dw_{og}}$, $\frac{dE}{d\theta_o}$ и $\frac{dE}{d\mu_{gi}}$ за всички входни стойности с индекс *i* и $\frac{dE}{d\sigma_g}$ при конкретен гаусов неврон с индекс *g*. При постъпващо ядро от тип *LearningOutputPattern* за текущото ядро се пресмята натрупаната грешка E_o за дадения изходен неврон. След това се съхранява като стойност за конкретния изходен неврон. Ако постигната грешка E_o е по-голяма от най-голямата достигната до момента $E^{max}{}_o$, то се записва новата максимална грешка с входните и изходните набори, при които е постигната. След приключване на изчисленията се инициализира новата желана изходна стойност за следващата итерация. С тази стъпка приключва работата на характеристичната функция *CharL1L45*, преди да премине ядрото на позиция L_5 за ядро от тип *LearningInputPattern* и L_6 за ядро от тип *LearningOutputPattern*. Характеристичната функция *CharL1L46* се активира при постъпване на ядра от тип *LearningInputPattern* или *LearningOutputPattern* на входно-изходната позиция *L*46. Пресмятанията в тази характеристична функция са еднакви с тези от *CharL1L45*, но се извършват следните ядра от тип *Learning-InputPattern* със следните ID-та: *LearningInputPatternNetTwoPrim*, *LearningInputPatternNetTwoSecond*, *LearningInputPatternNetTwoTerz* и ядро от тип *LearningOutputPattern* с ID *LearningOutputPatternNetTwo* на втората невронна мрежа.

За изпълнение в правилния ред на характеристичните функции, които са разгледани по-горе, е необходимо да се контролират разрешаването и приоритетът на преминаване на ядрата през прехода. Тези дейности се изпълняват от предикатите на прехода Z_1, както при всички останали преходи. Преходът Z_1 има девет предиката, а преминаването на ядрата от всички останали входни позиции към изходните такива се определят чрез константи на истинност. На фиг. 4.8. по-долу са представени всички предикати и константи на истинност, които определят преминаването към правилните изходни позиции за всички ядра, а в табл. 4.4. са описани формално всички предикати за прехода Z_1.

R_1	L_4	L_5	L_6	L_7	L_8	L_{45}	L_{46}
L_1	True	False	False	False	False	False	False
L_2	True	False	False	False	False	False	False
L_3	$W_{3,4}$	$W_{3,5}$	$W_{3,6}$	$W_{3,7}$	$W_{3,8}$	False	False
L_4	True	False	False	False	False	False	False
L_{27}	False	False	False	False	False	True	False
L_{44}	False	False	False	False	False	False	True
L_{45}	False	$W_{45,5}$	$W_{45,6}$	False	False	False	False
L_{46}	False	False	False	$W_{46,7}$	$W_{46,8}$	False	False

Фиг. 4.8. Матрица на предикатите R_1

Предикатът $W_{3,4}$ е един от първите предикати, които се изпълняват. Той разрешава преминаването на ядра, съхраняващи параметрите на двете невронни мрежи, в позиция L_4. Следващият предикат, който определя преминаването на ядра от входна позиция L_3, е $W_{3,5}$. $W_{3,5}$ разрешава преминаването на ядро от тип *LearningInputPattern*, което носи информация за входните въздействия, които са приложени към входовете на първата невронната мрежа. Същата проверка за втората невронна мрежа се изпълнява от предиката $W_{3,7}$. Предикатът $W_{3,6}$ пропуска ядро от тип *LearningOutputPatternNetOne*, носещо информация за желаните изходни резултати на изходния неврон за първата невронна мрежа.

Предикат	Описание
$W_{3,4}$	**((Type == NetworkParametersOne)|| (Type == NetworkParametersTwo))**
$W_{3,5}$	**(Type ==LearningInputPattern) & (Id == LearningInputPatternNetOnePrim)**
$W_{3,6}$	**(Type == LearningOutputPatternNetOne)**
$W_{3,7}$	**((Type ==LearningInputPattern) & ((Id == LearningInputPatternNetTwoPrim) || (Id == LearningInputPatternNetTwoSecond)|| (Id == LearningInputPatternNetTwoTerz))**
$W_{3,8}$	**(Type == LearningOutputPatternNetTwo)**
$W_{45,5}$	**(Type==LearningInputPattern)**
$W_{45,6}$	**(Type == LearningOutputPatternNetOne)**
$W_{46,7}$	**(Type==LearningInputPattern)**
$W_{46,8}$	**(Type == LearningOutputPatternNetTwo)**

Табл. 4.4. Описание на предикатите в R_1

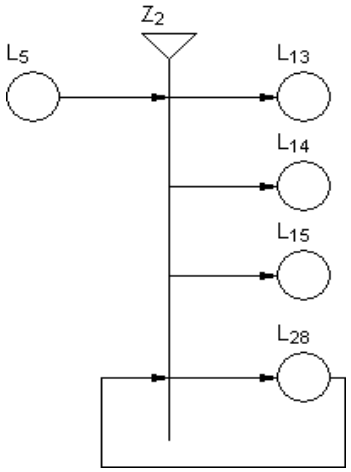
За втората невронна мрежа предикатът $W_{3,8}$ пропуска само ядрото *LearningOutputPatternNetTwo*, носещо също както $W_{3,6}$ желаните резултати за изходните неврони.

Преходът Z_1 изпълнява две основни функции от гледна точка на симулирането на невронната мрежа. Първата функция е да се генерират и инициализират подходящите ядра, така че да може да бъде симулирана дадена невронна мрежа. Втората функция, която изпълнява, е да обработи резултатите от обучението на всеки набор и от всяка епоха за дадена невронна мрежа. Трябва да се отбележи, че тези две дейности се извършват едновременно за всяка невронна мрежа, като се контролира и текущата грешка от обучението на всяка една.

4.2.2 Описание на преход Z_2

$Z_2 = <\{L_5, L_{28}\}, \{L_{13}, L_{14}, L_{15}, L_{28}\}, R_2, \wedge(L_5, L_{28}) >$

Преходът Z_2 разпределя ядрата от тип *LearningInputPattern* към съответния преход, който симулира работата на гаусов неврон в обобщената мрежа. За активирането на прехода е необходимо да са постъпили ядра на позиция L_5 и L_{28} в прехода. На позиция L_5 може да постъпят от едно до три ядра от тип *LearningInputPattern*. В конкретния случай, постъпва ядрото *LearningInputPatternNetOne*, което носи конкретния набор за обучение на невронната мрежа.



Фиг. 4.9. Графично представяне на прехода Z_2

Във входно-изходната позиция L_{28} цикли ядрото *BestNeuralNetworkOne*, което съхранява параметрите на най-добре обучената невронна мрежа до момента. Детайли за ядрата могат да бъдат видяни в табл. 4.5.

Преходът Z_2 има три характеристични функции – *CharL5L13*, *CharL5L14* и *CharL5L15*, които се грижат ядрата от типа *LearningInputPattern* да получат конкретните входни набори за обучение, които ще въздействат при обучението на невронната мрежа при определената итерация и епоха. Стъпките, които се извършват, са еднакви и при трите функции и затова е даден пример само с характеристичната функция *CharL5L13*. При постъпване на ядро *LearningInputPatternNetOne* в позиция L_5 и разрешаване

Ядро	Начални характеристики	Позиция на създаване/постъпване
LearningInput-PatternNetOne	*GaussNeuronId*, *Id*, *Type*, *Value*, *Inputs*	L_5
BestNeural-NetworkOne	*BestEpoch*, *NumberOfGaussNeurons*, *NumberOfMaxGaussNeurons*, *InputDimension*, *OutputDimension*, *OutputNeuronThreshold*	L_{28}

Табл. 4.5. Таблица на ядрата в преход Z_2

за преминаване към позиция L_{13} от предиката $W_{5,13}$ се активира характеристичната функция *CharL5L13*. Функцията прави проверка дали ядрото *LearningInputPatternNetOne* е преминало в позиция L_{13}. При постъпване на ядрото се нулират всички временни резултати от предишната итерация, след което се установява новата стойност на новата итерация. След пресмятането на новата итерация се присвоява и входният набор за обучение на характеристиката *Inputs*, който ще бъде използван от гаусовите неврони на следващия етап от обучението. С тази стъпка приключва изпълнението на характеристичната функция *CharL5L13*. Преходът Z_2 разполага с три предиката – $W_{5,13}$, $W_{5,14}$ и $W_{5,15}$. Всеки предикат разрешава преминаването на дадено ядро, притежаващо харакатеристика *Id* с определена стойност. Предикатът $W_{5,13}$ разрешава преминаването на ядро с *Id*, имащо стойност *LearningInputPatternNetOnePrim*, от позиция L_5 към позиция L_{13}. Работата на останалите два предиката е същата, но те разрешават преминаването на ядро с различна стойност към друга изходна позиция. На фиг. 4.10 е представена матрицата на преходите R_2 с всички предикати и константни булеви стойности между всеки два входни и изходни прехода. Формалното описание на предикатите $W_{5,13}$, $W_{5,14}$ и $W_{5,15}$ е представено на табл. 4.8 по-долу.

R_2	L_{13}	L_{14}	L_{15}	L_{28}
L_5	$W_{5,13}$	$W_{5,14}$	$W_{5,15}$	False
L_{28}	Falses	False	False	True

Фиг. 4.10. Матрица на предикатите R_2

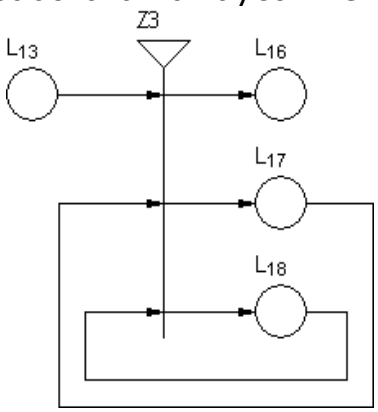
Предикат	Описание
$W_{5,13}$	**(Id == LearningInputPatternNetOnePrim)**
$W_{5,14}$	**(Id == LearningInputPatternNetOneSecond)**
$W_{5,15}$	**(Id == LearningInputPatternNetOneTerz)**

Табл. 4.8. Описание на предикатите в R_2

Преходът Z_2 може да бъде разглеждан като разпределител на ядрата, носещи входните въздействия на невронната мрежа към съответните преходи, които представят гаусовите й неврони. Преходът Z_2 има за двойник прехода Z_7, който извършва същата дейност за втората конкурентно обучавана невронна мрежа, поради което не е представен отделно.

4.2.3 Описание на преход Z_3

Преходът Z_3 симулира една от важните функции в процеса на обучение на невронната мрежа, а именно работата на гаусов неврон (вж. точка *4.1.2*).



Фиг. 4.11. Графично представяне на прехода Z_3

Z_3 = <{L_{13}, L_{17}, L_{18}}, {L_{16}, L_{17}, L_{18}}, R_3, $\vee(\wedge(L_{13}, L_{18}), L_{17})$ >

Тук се разглежда само работа на преход, който симулира гаусов неврон. Преходът Z_3 има три входни позиции L_{13}, $L17$ и L_{18}. В позиция L_{13} постъпва ядро *LearningInputPatternNetOne*. Входно-изходната позиция L_{18} съхранява ядрото *GaussNeuronOne*. В него се запазват всички параметри на гаусовия неврон, необходими за симулирането му. За да се активира преходът Z_3, е необходимо да са постъпили ядрата *LearningInput-PatternNetOne* и *GaussNeuronOne* в позиции L_{13} и L_{18}. В табл. 4.9 са представени двете ядра с техните характеристики.

Ядро	Начални характеристики	Позиция на създаване/постъпване
LearningInput-PatternNetOne	*GaussNeuronId*, *Id*, *Type*, *Value*, *Inputs*	L_{13}
GaussNeuronOne	*dMu*, *dSigma*, *Id*, *Mu*, *Sigma*, *NumberOfConnections*, *IsInitialized*	L_{18}

Табл. 4.9. Таблица на ядрата в преход Z_3

За да се симулира гаусов неврон, е необходимо да бъдат извършени определени изчисления върху характеристиките на невронната мрежа. Тези изчисления се извършват от характеристичните функции. В този преход има две такива характеристични функции – *CharL18L17* и *CharL18L18*. Характеристичната функция *CharL18L17* извършва пресмятанията на входните въздействия върху гаусовия неврон. Първата стъпка е да извлече входните набори от характеристиката *Inputs* на постъпилото ядро *LearningInputPatternNetOne*. След като са извлечени входните набори от ядрото *GaussNeuronOne*, се извлича характеритиката *Mu*. Тя съхранява математическото очакване за всеки един от входовете на невронната мрежа. От *GaussNeuronOne* се извлича също така и характеристиката *Sigma*, която съхранява стандартното отместване за гаусовия неврон. Чрез получените характеристики се изчислява функцията

$$\phi(v) = exp\left(-\frac{\|x-x_j\|^2}{2\sigma_j^2}\right) \quad (4.22)$$

за всяка входна стойност (x), както е описано в точка 4.1.2. В конкретния случай, параметърът x_j се замества с характеристиката *Mu*, σ се замества с характеристиката *Sigma*, а стойността x е стойност от вектора с входните въздействия. Чрез тези изчисления се извършва правият ход на обучението. Следващата стъпка от него е изчисляването на въздействието на гаусовите неврони върху изходните неврони. Характеристичната функция *CharL18L18* се активира при итеративното преминаване на ядрото *GaussNeuronOne* през входно-изходната позиция L_{18}. Целта на тази функция е да инициализира ядрото *GaussNeuronOne* с подходящите параметри от ядрото *NetworkParametersOne*.

За да работи правилно преходът Z_3, е необходимо да се определят позициите, през които да преминат ядрата. Редът на преминаване се определя от предикатите и булевите стойности от матрица на предикатите R_3. По-долу на фиг. 4.12 е представена матрицата R_3 с всички предикати и булеви константи. Булевите константи не представляват интерес от гледна точка промяната на поведението на прехода, затова не се разглеждат детайлно. Матрицата R_3 има един предикат $W_{18,18}$, който се използва да провери дали ядрото *GaussNeuronOne* е инициализирано.

Характеристична функция	Позиция на стартиране	Позиция на завършване	Използвани ядра
CharL18L17	L_{18}	L_{17}	*GaussNeuronOne*
CharL18L18	L_{18}	L_{18}	*GaussNeuronOne*, *Network-ParametersOne*

Табл. 4.11. Таблица на характеристичните функции на прехода Z_3

R_3	L_{16}	L_{17}	L_{18}
L_{13}	False	True	False
L_{17}	True	False	False
L_{18}	False	False	$W_{18,18}$

Фиг. 4.12. Матрица на предикатите R_3

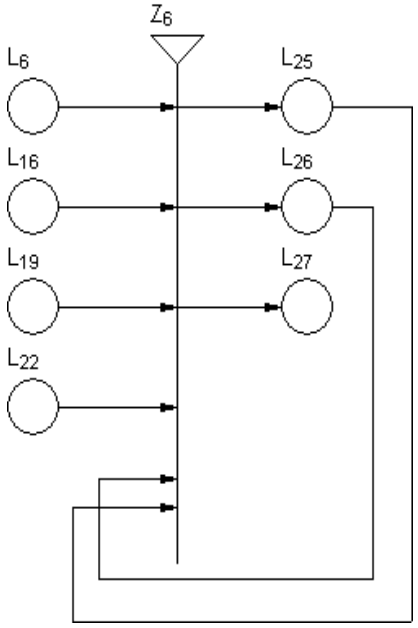
Предикат	Описание
$\boldsymbol{W_{18,18}}$	IsInitialized == True

Табл. 4.12. Описание на предикатите в R_3

Преходът Z_3 симулира работата на гаусов неврон от невронната мрежа. В обобщената мрежа симулацията на гаусов неврон се изпълнява още от преходите Z_6, Z_7 и Z_8. Тяхното описание не е необходимо, защото тези преходи са еднотипни и единствената разлика е в ядрата, с които работят.

4.2.4 Описание на преход Z_6

Преходът Z_6 симулира работата на изходен неврон на невронната мрежа. За правилното функциониране на прехода е необходимо да има постъпили ядра едновременно в позициите L_6, L_{16}, L_{19}, L_{22} и L_{26}.



Фиг. 4.13. Графично представяне на прехода Z_6

$$Z_6 = <\{L_6, L_{16}, L_{19}, L_{22}, L_{25}, L_{26}\}, \{L_{25}, L_{26}, L_{27}\}, R_6, \vee(\wedge(L_6, L_{16}, L_{19}, L_{22}, L_{26}), L_{25}) >$$

В прехода входните позиции са шест $\{L_6, L_{16}, L_{19}, L_{22}, L_{25}, L_{26}\}$. В позиции L_{16}, L_{19} и L_{22} са предвидени да постъпят ядрата от преходите, симулиращи гаусови неврони като Z_3. Входната позиция L_6 се използва от ядрото *LearningOutputPatternNetOne*, което носи желания изходен резултат при определеното входно въздействие. Позиция L_{25} е входно-изходна и се използва за синхронизация при постъпването на ядрата *LearningOutput-PatternNetOne* и *LearningInputPatternNetOnePrim*. Последната позиция е входно-изходната L_{26}, на която позиция е цикли ядрото *OutputNeuronOne*. Ядрото *OutputNeuronOne* съхранява всички параметри на симулирания изходен неврон. Изходната позиция L_{27} приема постъпилите ядра от позиция L_{25} *LearningInputPatternNetOnePrim* и *LearningOutputPatternNetOne*. По-долу в табл. 4.13 са представени всички ядра с техните характеристики.

Ядро	Начални характеристики	Позиция на създаване/постъпване
LearningInput-PatternNetOnePrim	*GaussNeuronId*, *Id*, *Type*, *Value*	L_{16}
LearningOutput-PatternNetOne	*Id*, *Type*, *Output*	L_6
OutputNeuronOne	*MaxGaussNeurons*, *dThr*, *dW*, *Error*, *IsInitialized*, *Output*, *Thr*, *W*, *Y*	L_{26}

Табл. 4.13. Таблица на ядрата в преход Z_6

За да симулира правилно работата на изходен неврон, преходът Z_6 трябва да използва характеристични функции вж. табл. 4.15 по-долу.

Характеристична функция	Позиция на стартиране	Позиция на завършване	Използвани ядра
CharL16L26	L_{16}	L_{26}	*OutputNeuronOne*, *NetworkPara-metersOne*
CharL16L27	L_{16}	L_{27}	*OutputNeuronOne*, *LearningInput-PatternNetOnePrim*, *Learning-Output-PatternNetOne*

Табл. 4.15. Таблица на характеристичните функции на прехода Z_6

В случая се използват две характеристични функции, които симулират поведението на изходен неврон. Първата от тях, която се изпълнява, е *CharL16L26*. Тя се извиква еднократно, за да инициализира тегловните коефициенти w_g, dw_g, θ и $d\theta$, където индексът *g* съответства на номера на гаусовия неврон. След инициализацията на основните параметри на ядрото *OutputNeuronOne*, преходът е готов за симулация на изходен неврон. При постъпване на ядрата *LearningInputPatternNetOnePrim* и *Learning-OutputPatternNetOne* в позиция L_{27} се активира характеристичната функция *CharL16L27*. Тази функция симулира активирането на изходния неврон и пресмятането на входните въздействия от гаусовите неврони. За да се изчисли резултатът на изходния неврон, при постъпване на неврон от тип *LearningInputPattern* се извлича стойността от характеристиката *Value*, която се умножава със съответния тегловен коефициент w_g в зависимост от *Id*-то на ядрото. Така полученият резултат се натрупва за всички входни ядра и се съхранява в характеристиката *Output* на *OutputNeuronOne*. След като изходният резултат е изчислен, следващите стъпки от процеса на обучение се правят в прехода Z_1 от вече натрупаните резултати от всички изходни неврони.

За правилната симулация на изходен неврон от прехода Z_6 е необходимо ядрата да постъпват по определен начин. Правилната последователност се контролира от матрицата на преходите R_6 с нейните предикати и булеви константи. По-долу на фиг. 4.14. е представена матрицата на преходите R_6 в табличен вид, а в табл. 4.16 са описани формално предикатите. За да изчисли характеристичната функция *CharL16L27* изходния резултат правилно, е необходимо ядрата *Learning-InputPatternNetOnePrim* и *LearningOutputPatternNetOne* да постъпят едновременно в позиция L_{25}. Това условие се управлява от предиката $W_{6,25}$. За да бъде изпълнено условието, е необходимо да се контролира преминаването на ядрото *LearningOutput-PatternNetOne* от позиция L_6 в позиция L_{25} едновременно с ядрото *LearningInputPatternNetOnePrim*, идващо от позиция L_{16}. Този факт налага употребата на предикат "*WasThere*" на симулационната среда върху ядрото *LearningInputPatternNetOnePrim*, което

разрешава преминаването на ядрото с желаните изходни резултати от позиция L_6 в позиция L_{25}.

R_6	L_{25}	L_{26}	L_{27}
L_6	$W_{6,25}$	False	False
L_{16}	True	False	False
L_{19}	True	False	False
L_{22}	True	False	False
L_{25}	False	False	True
L_{26}	False	True	False

Фиг. 4.14. Матрица на предикатите R_6

Предикат	Описание
$W_{6,25}$	**WasThere(**LearningInputPatternNetOnePrim**)**

Табл. 4.16. Описание на предикатите в R_6

Преходът Z_6 описва принципно функционирането на прехода Z_{11}, който симулира работата на изходен неврон за втората невронна мрежа и затова вторият преход не се описва самостоятелно. Преходът Z_6 е последният, който се изпълнява преди ядрата да се завърнат в позиция L_{45} на прехода Z_1. От позиция L_{45} се стартира процесът по натрупване на резултати или корекция на параметрите, както е описано в прехода Z_1.

4.3 Постигнати резултати

При симулация на така описания модел на невронната мрежа с радиални базисни функции са получени следните резултати, описани в табл. 4.17.

Епоха	Невронна мрежа 1	Невронна мрежа 2
1	0.039668	0.413347
2	0.01467	0.042485
3	0.007496	0.00227
4	0.004319	0.001184
5	0.002709	0.001042
6	0.001841	0.000904
7	0.001356	0.00076
8	0.001077	0.000612
9	0.000912	0.000467
10	0.00081	0.000331
11	0.000744	0.000213
12	0.000697	0.000121
13	0.000662	0.000057
14	0.000634	0.000022
15	0.00061	0.000006
16	0.000588	0.000001
17	0.000569	0
18	0.000551	0

Табл. 4.17. Грешка при обучението на двете невронни мрежи

От резултатите в таблицата може да се заключи, че използването на паралелна оптимизация върху повече невронни мрежи води до по-добри резултати, както се наблюдава при резултатите на втората невронна мрежа а времето за обучение може да бъде значително по-малко.

Предимствата на обобщеномрежовия модел са следните:

- Прилага се конкурентно управление на обучавани невронни мрежи;
- За първи път пресмятането на диференциалните уравнения е извършено в обобщената мрежа;
- Синхронизирани са конкурентните процеси на обучение на двете невронни мрежи;
- Симулирано е обучение на истинска невронна мрежа и са постигнати резултати с точност 4 порядъка след десетичния знак.

Предимства на алгоритъма за паралелна оптимизация на невронни мрежи чрез конкурентно обучение на две или повече невронни мрежи са:

- Неколкократно намаляване на времето за обучение, в зависимост от конкурентността на изчислителните ресурси;
- Постигане на същата точност на резултатите, както при последователно обучение на невронните мрежи;
- Максимално оползотворяване на наличните паралелни изчислителни ресурси;
- Възможност за допълнителни оптимизации на база на вече получените резултати от обучените невронни мрежи, в зависимост от типа на невронната мрежа.

Обобщеномрежовият модел от тази глава е публикуван в [13].

Глава 5. Обобщеномрежов модел на алгоритъм за конкурентно обучение на многослойни невронни мрежи

Цел на настоящата глава от дисертационния труд е да представи алгоритъм за конкурентно обучение на многослойни невронни мрежи. Така поставената цел изпълнява задача 4 от задачите на дисертационния труд.

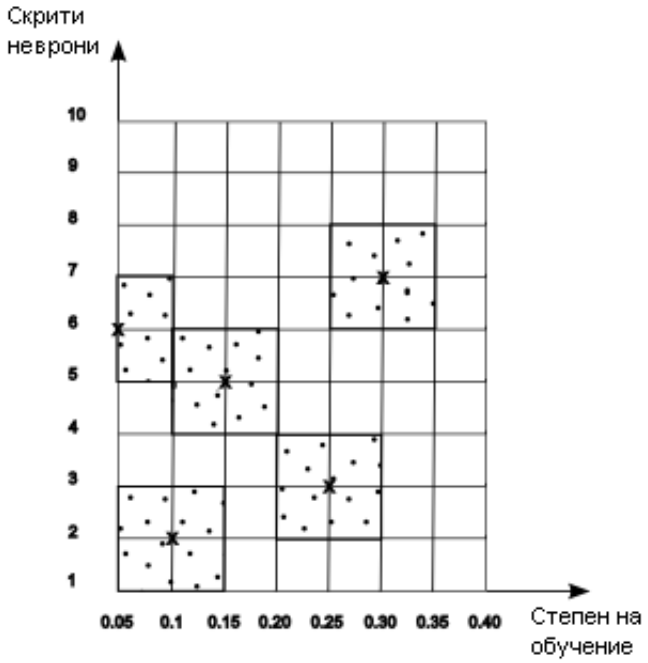
5.1 Алгоритъм за конкурентно обучение на многослойни невронни мрежи

Всички невронни мрежи притежават независими параметри, които могат да бъдат оптимизирани с цел намиране на най-добре обучените невронни мрежи спрямо получените резултати. Предложеният алгоритъм прилага подход на конкурентно търсене върху поле на невронна мрежа. Под поле на невронна мрежа се разбира двумерно евкилидово пространство с два независими обучителни параметъра на невронна мрежа. В случая ще бъде представен алгоритъм за паралелна обработка на невронни мрежи с два значими параметъра за оптимизация. Възможно е да се използват повече от два параметъра, но това изследване е предмет на бъдеща работа.

Избраният алгоритъм с два оптимизационни параметъра описва двуфазен подход за намиране на най-добре обучената невронна мрежа. Първата фаза е да се направи отсяващ подбор на най-добре обучените невронни мрежи, а втората фаза е да се извърши селективен подбор на най-добре обучената невронна мрежа в областите около вече подбраните невронни мрежи. Най-ефективният начин да се изберат оптимизационни параметри за обучение е като се определят параметрите с най-голямо влияние върху процеса на обучение. В случая, това е възможно да бъде направено с вид факторен анализ за параметрите с най-голямо влияние. Нека бъде прието, че параметрите за обучение с най-голямо влияние са избрани и те са броят на скритите неврони (h) и степента на обучение (η). Първо трябва да се определи минималната стъпка на увеличение или намаление за всеки един от тези параметри. Минималната стъпка на увеличение на скритите неврони се отбелязва с δh, а минималната стъпка на увеличение на степента на обучение се отбелязва с $\delta\eta$. Максималната стъпка на увеличение се представя с Δh за скритите неврони и с $\Delta\eta$ за степента на обучение. Смисълът на минималната стъпка е число, което е лесно за пресмятане и значимо за процеса на обучение. Стойности, по-малки от минималната стъпка, нямат влияние върху резултата от процеса на обучение. Относно максималната стъпка за увеличение е важно да се дефинира горна граница, която би помогнала да се прескочи някакъв вид локален екстремум на една добре обучена невронна мрежа. Изборът на тези стъпки е определящ за крайния резултат от търсенето на най-добре обучена невронна мрежа. След избора на минимална и максимална стъпка за степента на обучение и скритите неврони е необходимо да се избере нормална стъпка за всеки от параметрите на полето. Означението на нормалната стъпка за степен на обучение е η_n, а на скритите неврони е h_n. Стойностите на нормалните стъпки са онези, които се използват в нормалния обучителен процес на всяка невронна мрежа. Също така е важно да се определи минимална и максимална стойност за значимите параметри, което ще дефинира област на търсене от полето за най-добро решение. С h_{min} и h_{max} се отбелязват минималният и максималният брой скрити неврони, а с η_{min} и η_{max} се отбелязват минималната и максималната степен на обучение. Следващата стъпка е да се изберат начални точки за стартиране на процеса на обучение за различни невронни мрежи. Изборът на начални точки зависи от минималните и максималните стойности на значимите параметри за невронната мрежа. Нека се приеме, че $h_{min} = 1$ и $h_{max} = 10$ неврона. Ако $h_n = 2$, тогава $h \in \{1, 3, 5, 7, 9\}$ и пет невронни мрежи могат да започнат процес на конкурентно обучение. Ако се приеме, че минималната степен на обучение е $\eta_{min} = 0.05$, а максималната такава е $\eta_{max} = 0.35$, то може да се избере минималната степен на обучение $\delta\eta = 0.02$, а за нормална стъпка на обучение $\eta_n = 0.5$. Избраните граници за степента на обучение създава множество от степени $\eta \in \{0.05, 0.1, ..., 0.35\}$. Следователно, могат да се използват седем стъпки на обучение върху пет избрани невронни мрежи, които са определени от броя на скритите неврони. Броят на степените на обучение и скритите неврони показва, че е необходимо да бъдат обучени 35 невронни мрежи. Те ще бъдат

класифицирани като добре обучени невронни мрежи или лошо обучени такива според параметъра ***ε.*** Параметърът ***ε*** представлява възможната грешка, измерена в края на процеса на обучение. Оттук следва, че всички невронни мрежи, чиято грешка в края на обучителния процес е по-малка от ***ε***, се считат за добре обучени.

След като процесът на конкурентно обучаваните невронни мрежи е завършен с по-горе описаните параметри, добре обучените невронни мрежи се отделят от всички обучавани невронни мрежи и се използват като центрове за следващата фаза на процеса на обучение в търсене на най-добре обучената невронна мрежа. Пример за вече определени центрове след завършване на първата фаза е даден на фиг. 5.1 по-долу.

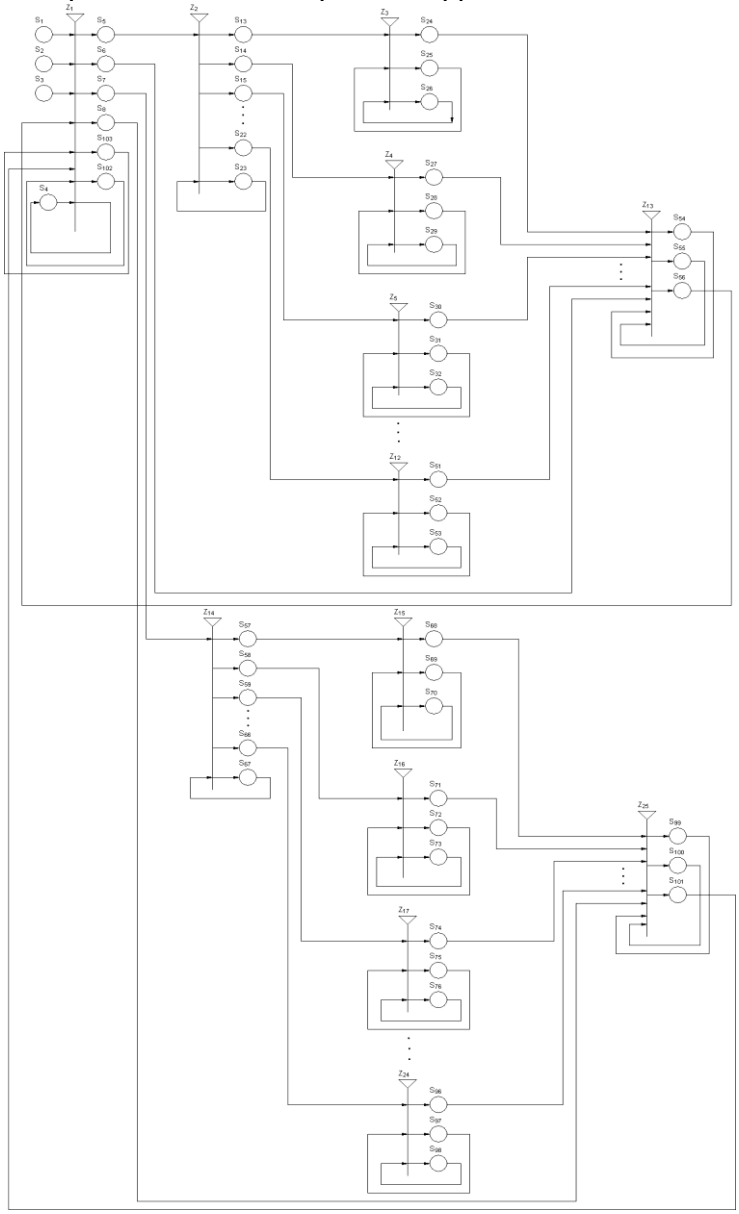


Фиг. 5.1. Поле на невронна мрежа

За следващата фаза се избира симетричен интервал около всеки един от центровете. В разглеждания случай се избират два скрити неврона за симетричен интервал около центъра и степен на обучение 0.1 около всеки от тях. Дефиницията за център е наредена двойка от координати в полето, върху която е поставена добре обучена невронна мрежа. В този момент процесът на конкурентно обучение на невронни мрежи започва да използва по-малка степен на обучение, която в този случай е $\delta\eta = 0.02$, и по-малка стъпка за скритите неврони $h_{min} = 1$. Конкурентният процес на обучение се стартира отново с нови параметри около всеки център и с по-малки стойности на всеки значим параметър. Събраната информация за най-добре обучената невронна мрежа за всеки център се съхранява до края на процеса на обучение и се избира най-добре обучената невронна мрежа с минимална грешка.

5.2 Обобщеномрежов модел на алгоритъма за конкурентно обучение на многослойни невронни мрежи

Представеният ОМ модел на алгоритъма за конкурентно обучение на многослойни невронни мрежи на фиг. 5.2. по-долу е редуциран. Той не притежава темпорални компоненти, приоритети на преходите и няма ограничения в капацитета на позициите и дъгите.



Фиг. 5.2. Обобщеномрежов модел на алгоритъм за конкурентно обучение на многослойни невронни мрежи

Състои се от двадесет и пет прехода :

$A = \{Z_1, Z_2, Z_3, Z_4, Z_5, Z_6, Z_7, Z_8, Z_9, Z_{10}, Z_{11}, Z_{12}, Z_{13}, Z_{14}, Z_{15}, Z_{16}, Z_{17}, Z_{18}, Z_{19}, Z_{20}, Z_{21}, Z_{22}, Z_{23}, Z_{24}, Z_{25}\}$,

където преходите описват следните процеси:

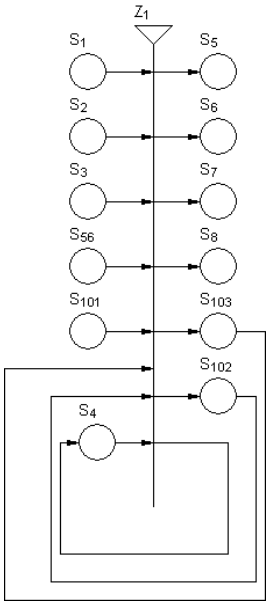
Z_1 – Формиране на начални условия и структура на невронните мрежи, съхраняване на основните параметри на алгоритъма за конкурентно обучение на многослойни невронни мрежи, генериране на нови невронни мрежи в зависимост от фазата и текущите стойности на параметрите на алгоритъма за конкурентно обучение;

Z_2, Z_{14} – Насочват ядрата с входното въздействие към съответните преходи, симулиращи скрити неврони;

Z_3, Z_4, Z_5, Z_6, Z_7, Z_8, Z_9, Z_{10}, Z_{11}, Z_{12}, Z_{14}, Z_{15}, Z_{16}, Z_{17}, Z_{18}, Z_{19}, Z_{20}, Z_{21}, Z_{22}, Z_{23}, Z_{24} – Изчисляване на входното въздействие върху скритите неврони и пренасяне на резултатите към преходите, представящи изходите на невронната мрежа;

Z_{13}, Z_{25} – Изчисляване на резултата на изходите на невронната мрежа.

5.2.1 Описание на преход Z_1



Фиг. 5.3. Графично представяне на преход Z_1

$Z_1 = <\{S_1, S_2, S_3, S_4, S_{56}, S_{101}, S_{102}, S_{103}\}, \{S_4, S_5, S_6, S_7, S_8, S_{102}, S_{103}\}, R_1, \vee(\wedge(S_1, S_2, S_3, S_4), S_{56}, S_{101}, S_{102}, S_{103})>$

Преходът Z_1 изпълнява стъпките на алгоритъма за конкурентно обучение по отношение на създаването на невронните мрежи, част от обучението и натрупването на резултати от обучението на всяка една невронна мрежа. За активиране на прехода Z_1 е необходимо в позиции S_1, S_2, S_3 и S_4 да постъпят съответните ядра. В позиция S_1 постъпва ядро *ι*. Ядрото *ι* съхранява в себе си входните набори за обучение на невронната мрежа. Позиция S_2 приема ядро *o*. То носи съответно изходните набори за обучение на невронната мрежа. Причината за разделеното постъпване е по-лесната и едновременна обработка при генерирането на ядра за новосформираните невронни мрежи, стъпвайки на принципа "Copy-on-write" [129]. В позиция S_3 постъпват ядрата *η′* и *η″*. Тези ядра съхраняват данните за основните структури на двете невронни мрежи, които ще се обучават конкурентно. На тази позиция също така постъпват и ядрата *υ′* и *υ″*, които съхраняват информацията за желаните изходни резултати след завършване

на обучението, предстоящия изходен набор за обучение и типа на ядрото. Позиция S_4 приема ядра λ', λ'', ε' и ε''. Ядрата λ' и λ'' се ползват за съхранение на всички параметри, свързани с обучение на двете конкурентно обучавани мрежи. Ядрата ε' и ε'' служат за съхраняване на оценъчните параметри или различни типове грешки, в зависимост спрямо какво са пресметнати. Трябва да се отбележи, че алгоритъмът е предвиден за различни типове многослойни мрежи и понякога параметрите, с които работи, са различни в зависимост от алгоритъма за обучение на невронната мрежа.

В табл. 5.1 по-долу е представена таблица с описание на ядрата и основните характеристики, необходими за представяне на конкурентния алгоритъм за обучение на многослойни невронни мрежи. В преход Z_1 има няколко характеристични функции, които обработват новопостъпилите ядра. Ядрата в прехода могат да постъпят по два начина – при първоначално активиране на обобщената мрежа или при постъпване на ядра след поредната итерация от конкурентното обучение на невронните мрежи.

Ядро	Начални характеристики	Позиция на създаване/ постъпване	Описание на ядрото
I	$x_0^{\iota}, x_1^{\iota}, x_2^{\iota}, x_3^{\iota}$	S_1	Съхранява данни за входните неврони
o	$x_0^{o}, x_1^{o}, x_2^{o}, x_3^{o}$	S_2	Съхранява данни за изходните неврони
η′, η″	$x_0^{\eta}, x_1^{\eta}, x_2^{\eta}, x_3^{\eta}, x_4^{\eta}, x_5^{\eta}, x_6^{\eta}, x_7^{\eta}, x_8^{\eta}, x_9^{\eta}, x_{10}^{\eta}$	S_3	Съхранява данни за структурата на невронната мрежа
υ′, υ″	$x_0^{\upsilon}, x_1^{\upsilon}, x_2^{\upsilon}$	S_3	Съхранява данни за желаните изходни стойности на невронната мрежа
ξ′,ξ"	$x_0^{\xi}, x_1^{\xi}, x_2^{\xi}, x_3^{\xi}, x_4^{\xi}$	S_3	Съхранява данни за входните стойности на невронната мрежа
λ′, λ″	$x_0^{\lambda}, x_1^{\lambda}$	S_4	Съхранява данни за параметрите на обучение за дадена невронна мрежа
ε′, ε″	x_0^{ε}	S_4	Съхранява данни за параметричните грешки в процеса на обучение

Табл. 5.1. Таблица на ядрата в преход Z_1

В този случай не е необходимо да бъдат създавани отделни характеристични функции, които да обработят данните от постъпилите ядра, а се използват едни и същи.

Има два типа характеристични функции в прехода. Първият тип функции се активират при изпълнение на всяко обучение на набор, а вторият тип - при завършване на обучението на всички набори, т.е. епоха.

При постъпване на ядра η', υ' и ξ' на входна позиция S_3 или υ' и ξ' на входна позиция S_{102} се активира характеристичната функция *charProcess-IterationDataPrim*. Тази характеристична функция се използва за обработка и корекция на обучаващите параметри на първата невронната мрежа на всяка итерация от обучението за всяка *n*-торка на входния набор. За да се изпълни тази характеристична функция, е необходимо да е постъпило ядрото ξ' или някое генерирано от него ядро в позиция S_3 или S_{102}. Допълнително генерираните от ядрото ξ' ядра се определят от алгоритъма за конкурентно обучение според броя на скритите неврони в дадената невронна мрежа (вж. т. 5.1). Характеристичната функция *charProcess-IterationDataSecond* изпълнява подобни операции за втората невронна мрежа при постъпване на ядра η'', υ'' и ξ'' на позиция S_3 или на ядрата υ" и ξ'' на позиция S_{103}. Поведението на двете характеристични функции се различава според това какви са параметрите на дадената невронна мрежа за обучение и реда на стъпките, които трябва да бъдат изпълнени. Характеристичната функция, която се активира при завършване на обучението на първата невронна мрежа в една епоха е *charProcessEpochDataPrim*, което се случва при постъпване на ядра на позиция S_{102}. Предназначението на *charProcessEpochDataPrim* е да нанесе корекции на всички параметри, които подлежат на корекция, например като тегловните коефициенти, степента на обучение, различните коригиращи параметри, изчислени спрямо грешката на мрежата, и т.н. Ако процесът на обучение е приключен за дадената мрежа, то характеристичната функция се грижи за създаването на нова невронна мрежа спрямо изискванията на алгоритъма за конкурентно обучение на многослойни невронни мрежи. Създаването на нова невронна мрежа изисква:

- презареждане на новите параметри за ядрата, които участват в обучението;
- генериране на нови ядра, които се изискват за създаването на новата невронна мрежа;
- унищожаване на вече ненужни ядра с характеристики от старата невронна мрежа.

Характеристичната функция *charProcessEpochDataSecond* изпълнява същите инструкции като *charProcessEpochDataPrim*, като се съобразява с конкретните параметри на втората невронна мрежа. Има разграничение по пресмятане на нови стойности на параметри, генериране на нови ядра и сливане на ядра между двете симулирани невронни мрежи. Причината да има такова разделение е конкурентността на пресмятане, която предполага намаляване времето на изчакване за използване на дадена характеристика или ядро. След първоначалното преминаване на ядрата от прехода Z_1 през всички останали преходи, те постъпват отново в прехода Z_1. В този момент при постъпване на ядра от типа *LearningInputPattern* и *LearningOutput-Pattern*, например υ' и ξ', на позиция S_{56} се активира характеристична функция *charPrepareProcessIterationDataPrim*. Тази характеристична функция се използва, за да изчисли временните резултати в края на итерацията, да съхрани текущите характеристики и резултати за по-

нататъшна корекция на тегловните коефициенти, праговите коефициенти и други параметри след всяка епоха. Характеристичната функция *charPrepareProcessIterationDataSecond* се активира на позиция S_{101}, като изпълнява същия алгоритъм, както предишната функция, но за втората симулирана невронна мрежа. Характеристичната функция *charGenerateLearningTokens* се активира на позиция S_3. Тя служи за първоначалното генериране на всички ядра от типа *LearningInputPattern* и *LearningOutputPattern* при постъпване на позиция S_3. На позиция S_3 постъпва по едно ядро от всеки тип ξ', ξ'', υ' и υ''. На база първата стъпка от алгоритъма за конкурентно обучение се взима решение дали ядрата да бъдат разцепени, според нуждата за повече такива, според нуждата на алгоритъма за увеличаване броя на скритите неврони и изходни неврони, и т.н. За по-структурирана информация за позициите на активиране и завършване на всяка характеристична функция вж. табл. 5.3 по-долу. Преминаването на ядрата от входните към изходните позиции на прехода Z_1 се определя от матрицата на преходите R_1, която е представена в таблица на фиг. 5.4. Както във всяка матрица, така и в тази има константно зададени разрешения и забрани за преминаване между позициите.

Характеристична функция	Позиция на стартиране	Позиция на завършване	Използвани ядра
charProcessIteration-DataPrim	S_3/S_{102}	S_5	ξ'
charProcessEpoch-DataPrim	S_{102}	S_6	υ'
charProcessIteration-DataSecond	S_3/S_{103}	S_7	$\xi''_1, \xi''_2, \xi''_3$
charProcessEpoch-DataSecond	S_{103}	S_8	υ''
charPrepareProcess-IterationDataPrim	S_{56}	S_{102}	υ', ξ'
charPrepareProcess-IterationDataSecond	S_{101}	S_{103}	υ'', ξ''
charGenerateLearningTokens	S_3	$S_5/S_6/S_7/S_8$	$\xi', \xi''_1, \xi''_2, \xi''_3, \upsilon', \upsilon''$

Табл. 5.3. Таблица на характеристичните функции на прехода Z_1

Тези константи не носят значима информация при работата на прехода. Матрицата на преходите Z_1 има девет важни предиката $W_{3,4}$, $W_{3,5}$, $W_{3,6}$, $W_{3,7}$, $W_{3,8}$, $W_{45,5}$, $W_{45,6}$, $W_{46,7}$ и $W_{46,8}$, от които зависи взимането на решение за поведението на целия обобщеномрежов модел. Първият предикат $W_{3,4}$ разрешава преминаването на всички онези ядра, които са от типа *Network-Parameters*. Такива ядра са η' и η''. Предикатът $W_{3,5}$ проверява дали типът на ядрото, което се опитва да премине от позиция S_3 към позиция S_5, е *LearningInputPattern* и дали е предвидено за първата невронна мрежа. $W_{3,6}$ предикатът позволява преминаването на ядра от типа *LearningOutputPattern*, предвидени за употреба в първата невронна мрежа. $W_{3,7}$ е предикатът, отговарящ за насочване на ядра от типа *LearningInputPattern*, които са предвидени за втората невронна мрежа, от

позиция S_3 към позиция S_7. Този предикат е подобен на $W_{3,5}$ с разликата, че насочва ядрата, предвидени за втората невронна мрежа. Последният предикат, който разрешава преминаването на ядра от входната позиция S_3 към изходните позиции, е $W_{3,8}$. Той разрешава преминаването на ядра от типа *LearningOutputPattern*, които са предвидени за втората невронна мрежа. Предикатът $W_{3,8}$ изпълнява същата дейност като предиката $W_{3,6}$ с единствената разлика, че разрешава преминаването на ядра само за втората невронна мрежа. Формалното описание на предикатите е направено в табл. 5.4 по-долу.

R_1	S_4	S_5	S_6	S_7	S_8	S_{102}	S_{103}
S_1	True	False	False	False	False	False	False
S_2	True	False	False	False	False	False	False
S_3	$W_{3,4}$	$W_{3,5}$	$W_{3,6}$	$W_{3,7}$	$W_{3,8}$	False	False
S_4	True	False	False	False	False	False	False
S_{56}	False	False	False	False	False	True	False
S_{101}	False	False	False	False	False	False	True
S_{102}	False	$W_{102,5}$	$W_{102,6}$	False	False	False	False
S_{103}	False	False	False	$W_{103,7}$	$W_{103,8}$	False	False

Фиг. 5.4. Матрица на предикатите R_1

Следващата група от предикати е свързана с входната позиция S_{102}. Предикатът $W_{102,5}$ разрешава преминаването на ядра от типа *LearningInputPattern* към позиция S_5. Следващият предикат $W_{102,6}$, който се активира на позиция S_{102}, пропуска към позиция S_6 онези ядра, които са от тип *LearningOutputPattern*. Предикатът $W_{103,7}$ позволява преминаването на ядра от тип *LearningInputPattern* от входната позиция S_{103} към изходната позиция S_7. Този предикат оперира по същия начин, както предикатът $W_{102,5}$. Единствената разлика между тях е, че единият обслужва първата невронна мрежа, а вторият обслужва втората. По същия начин, предикатът $W_{103,8}$ разрешава преминаването на ядра от тип *LearningOutputPattern* от позиция S_{103} към позиция S_8, както предикатът $W_{102,6}$.

В обобщение, преходът Z_1 е предназначен за изпълнение на алгоритъма за конкурентно обучение на многослойни невронни мрежи от една страна, а от друга изпълнява част от операциите по конкурентното обучение на текущите две невронни мрежи.

Предикат	Описание
$W_{3,4}$	Ако ядрото е от тип *NetworkParameters* (η' || η''), премини на позиция S_4.
$W_{3,5}$	Ако ядрото е от тип *LearningInputPattern* и се индентифицира с ξ'_x, където $x \in [1,n]$, премини на позиция S_5.
$W_{3,6}$	Ако ядрото е от тип *LearningOutputPattern* и се индентифицира с u'_x, където $x \in [1,n]$, премини на позиция S_6.
$W_{3,7}$	Ако ядрото е от тип *LearningInputPattern* и се индентифицира с ξ''_x, където $x \in [1,n]$, премини на позиция S_7.

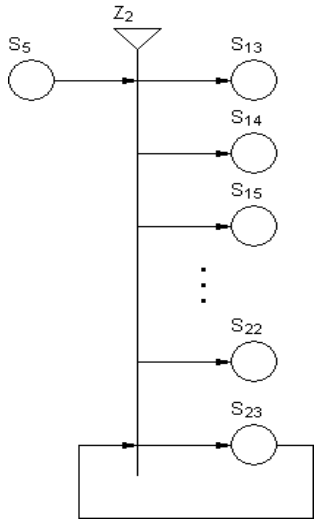
$W_{3,8}$	Ако ядрото е от тип *LearningOutputPattern* и се индентифицира с u''_x, където $x \in [1,n]$, премини на позиция S_8.
$W_{102,5}$	Ако ядрото е от тип *LearningInputPattern*, премини на позиция S_5.
$W_{102,6}$	Ако ядрото е от тип *LearningOutputPattern*, премини на позиция S_6.
$W_{103,7}$	Ако ядрото е от тип *LearningInputPattern*, премини на позиция S_7.
$W_{103,8}$	Ако ядрото е от тип *LearningOutputPattern*, премини на позиция S_8.

Табл. 5.4. Описание на предикатите в R_1

5.2.2 Описание на преход Z_2

$Z_2 = <\{S_5, S_{23}\}, \{S_{13}, S_{14}, S_{15}, S_{16}, S_{17}, S_{18}, S_{19}, S_{20}, S_{21}, S_{22}, S_{23}\}, R_2, \wedge(S_5, S_{23})>$

Преходът Z_2 разпределя всички постъпили ядра от тип *LearningInputPattern* към съответните преходи, симулиращи скрити неврони за конкретната невронна мрежа. Активирането на прехода Z_2 се определя от условието на прехода $\square = \wedge(S_5, S_{23})$.



Фиг. 5.5. Графично представяне на прехода Z_2

За да се активира преходът, е небходимо на позициите S_5 и S_{23} да са постъпили съответните ядра. На позиция S_5 трябва да постъпи поне едно ядро от типа *LearningInputPattern*, например ξ'. В позиция S_{23} постъпва ядрото β'. То съхранява основните параметри на най-добре обучената невронна мрежа до момента. В табл. 5.5 по-долу таблично са представени ядрата, които са използвани в прехода. Преходът Z_2 има няколко характеристични функции от типа *charPrepare-ForCalculationsForNextTransition*. Характеристичните функции от този тип се активират при постъпване на ядра в позиция S_5. Всички те отговарят за увеличаване на брояча на итерации в рамките на една епоха.

Ядро	Начални характеристики	Позиция на създаване/постъпване	Описание на ядрото
β′	$x_0^{\beta}, x_1^{\beta}, x_2^{\beta}, x_3^{\beta}, x_4^{\beta}, x_5^{\beta}$	S_{23}/S_{23}	Съхранява параметрите на най-добре обучената невронна мрежа.
$\xi_1' \dots \xi_n'$	$x_0^{\xi}, x_1^{\xi}, x_2^{\xi}, x_3^{\xi}, x_4^{\xi}$	$S_5/S_{13}...S_{23}$	Съхранява данни за входните стойности на невронната мрежа

Табл. 5.5. Таблица на ядрата в преход Z_2

Също така, тези характеристични функции се грижат за зареждане на съответния входен набор за обучение за дадения скрит неврон. Даденият по-долу пример се отнася за първата характеристична функция *charPrepareForCalculationsForNextTransitionOne*, а за останалите начинът на обработка е аналогичен. При постъпване на ядро ξ_1' в позиция S_5 характеристичната функция се активира, като първо извлича текущата итерация за първата невронна мрежа – от λ' увеличава стойността с единица и я записва отново в същата характеристика. След това се изтегля входен набор за обучение с индекс текущата итерация от ядрото *I* и се записва като характеристика в ядрото ξ_1'. По този начин всяка една характеристична функция, която се активира на позиция S_5, се активира и съхранява характеристиките за новия входен набор за обучение за конкретното ядро от типа ξ_n', където $n \in [2,n]$.

Характеристична функция	Позиция на стартиране	Позиция на завършване	Използвани ядра
charPrepareForCalculationsForNextTransitionOne	S_5	S_{13}	ξ_1'
charPrepareForCalculationsForNextTransitionTwo	S_5	S_{14}	ξ_2'
...	...	...	...
charPrepareForCalculationsForNextTransitionTen	S_5	S_{22}	ξ_n'

Табл. 5.7. Таблица на характеристичните функции на прехода Z_1

В табл. 5.7 е представена таблица с описание на характеристичните функции на прехода Z_2. За да премине всяко ядро от входна към изходна позиция в прехода Z_2, е необходимо разрешение от предикат или булева константа, която се съдържат в матрицата на преходите R_2. Тя е представена в таблица на фиг. 5.6 по-долу. Първият предикат е $W_{5,13}$, който разрешава преминаването на ядро от тип *LearningInputPattern*, предвиден за прехода, симулиращ първия скрит неврон на първата невронна мрежа Z_3. Предикатите $W_{5,14}$, $W_{5,15}$, $W_{5,16}$, $W_{5,17}$, $W_{5,18}$, $W_{5,19}$, $W_{5,20}$, $W_{5,21}$ и $W_{5,22}$ изпълняват същата проверка за съответния скрит неврон от първата невронна мрежа.

R_2	S_{13}	S_{14}	S_{15}	S_{16}	S_{17}	S_{18}	S_{19}	S_{20}	S_{21}	S_{22}	S_{23}
S_5	$W_{5,13}$	$W_{5,14}$	$W_{5,15}$	$W_{5,16}$	$W_{5,17}$	$W_{5,18}$	$W_{5,19}$	$W_{5,20}$	$W_{5,21}$	$W_{5,22}$	False
S_{23}	False	False	False	False	False	False	False	False	False	False	True

Фиг. 5.6. Матрица на предикатите R_2

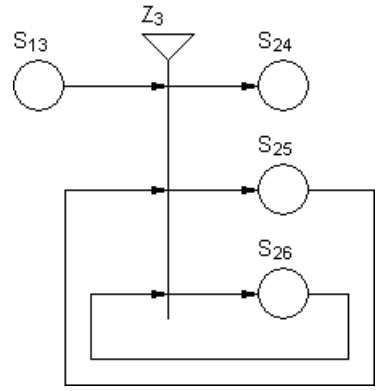
В табл. 5.8 е представено формално описание на предикатите от матрицата на предикатите R_2.

Предикат	Описание
$W_{5,13}$	Ако ядрото е от тип *LearningInputPattern* && x_0^{ξ}=0 на позиция S_{13}
$W_{5,14}$	Ако ядрото е от тип *LearningInputPattern* && x_0^{ξ}=1 на позиция S_{14}
$W_{5,15}$	Ако ядрото е от тип *LearningInputPattern* && x_0^{ξ}=2 на позиция S_{15}
$W_{5,16}$	Ако ядрото е от тип *LearningInputPattern* && x_0^{ξ}=3 на позиция S_{16}
$W_{5,17}$	Ако ядрото е от тип *LearningInputPattern* && x_0^{ξ}=4 на позиция S_{17}
$W_{5,18}$	Ако ядрото е от тип *LearningInputPattern* && x_0^{ξ}=5 на позиция S_{18}
$W_{5,19}$	Ако ядрото е от тип *LearningInputPattern* && x_0^{ξ}=6 на позиция S_{19}
$W_{5,20}$	Ако ядрото е от тип LearningInputPattern && x_0^{ξ}=7 на позиция S_{20}
$W_{5,21}$	Ако ядрото е от тип LearningInputPattern && x_0^{ξ}=8 на позиция S_{21}
$W_{5,22}$	Ако ядрото е от тип *LearningInputPattern* && x_0^{ξ}=9 на позиция S_{22}

Табл. 5.8. Описание на предикатите в R_2

Преходът Z_2 чрез своите характеристични функции и предикати е предназначен да съхранява новите набори за обучение в съответните ядра и да насочва ядрата към съответните преходи от Z_3 до Z_{13}. Преходите от Z_3 до Z_{13} симулират работата на скритите неврони в мрежата. Преходът Z_{14} е еднотипен на прехода Z_2 и симулира същото подготвяне на ядрата с набори и разпределяне към съответните неврони за втората невронна мрежа, поради което не е описан отделно в представянето на модела.

5.2.3 Описание на преход Z_3



Фиг. 5.7. Графично представяне на прехода Z_3

$Z_3 = <\{S_{13}, S_{25}, S_{26}\}, \{ S_{24}, S_{25}, S_{26}\}, R_2, \vee(\wedge(S_{13}, S_{26}), S_{25})>$

Преходът Z_3 симулира работата на скрит неврон от невронната мрежа. Първоначално преходът пресмята входните въздействия от всички входни неврони, които пристигат с ядрото ξ'_1, в характеристиката x_4^{ξ}. След пресмятането на входното въздействие, резултатът се използва като входен параметър за активиращата функция. Второто действие е пресмятането на резултата от активиращата функция. Преходът Z_3 има три входни (S_{13}, S_{25}, S_{26}) и три изходни позиции (S_{24}, S_{25}, S_{26}). Условието на прехода за активиране е □ = ∨(∧(S_{13}, S_{26}), S_{25}). Следователно, за да се активира преходът, е необходимо да са постъпили две ядра в позициите S_{13} и S_{26}. Във входната позиция S_{13} първоначално постъпва ядрото ξ'_1, а в позицията S_{26} постъпва ядрото χ'_1. След постъпването на тези две ядра в съответните позиции, преходът Z_3 се активира. В табл. 5.9 по-долу са описани ядрата на прехода Z_3.

Ядро	Начални характеристики	Позиция на създаване/постъпване	Описание на ядрото
ξ'_1	$x_0^{\xi}, x_1^{\xi}, x_2^{\xi}, x_3^{\xi}$	S_{13}	Съхранява данни за входните стойности на невронната мрежа
χ'_1	$x_0^{\chi}, x_1^{\chi}, \ldots, x_n^{\chi}$	S_{26}	Съхранява всички параметри на скрит неврон

Табл. 5.9. Таблица на ядрата в преход Z_3

Характеристичните функции на прехода, които изчисляват крайния резултат на скрития неврон, са *charInitHiddenNeuron* и *charDoCalculations*. Функцията *charInitHiddenNeuron* се активира при първоначално постъпване на ядрото χ'_1 на позиция S_{26}. Тя инициализира началните параметри на ядрото χ'_1, които са необходими за последващата работа и обучение на дадения скрит неврон. Всички характеристики на скрития неврон се съхраняват в ядрото χ'_1, което цикли на входно-изходната позиция S_{26}. След като постъпи ядрото на позиция S_{13}, то преминава на входно-изходната позиция S_{25}. В позиция S_{25} се активира характеристична функция *charDoCalculations* при постъпване на ядрото ξ'_n. Характеристичната функция извлича всички входни въздействия на входните неврони от постъпилото ядро и пресмята входното въздействие със съответните тегловни коефициенти от ядрото χ'_1. След това се изчислява и резултатът от активиращата функция. Последната използва резултата от входното въздействие и генерира изходен резултат, който бива записан в характеристика на ядрото ξ'_n. Типът на активиращата функция се определя от алгоритъма за обучение на многослойните невронни мрежи.

Характеристична функция	Позиция на стартиране	Позиция на завършване	Използвани ядра
charDoCalculations	S_{25}	S_{24}	ξ'_n
charInitHiddenNeuron	S_{26}	S_{26}	χ'_1

Табл. 5.11. Таблица на характеристичните функции на прехода Z_3

В табл. 5.11 по-долу са представени с формално описание основните характеристики на характеристичните функции за прехода Z_3. Преходът Z_3 има матрица на преходите R_3, която съдържа само един предикат (фиг. 5.8). Предикатът на прехода е $W_{26,26}$. Той проверява дали ядрото χ_1' е инициализирано при всяко преминаване през входно-изходната позиция S_{26}.

R_3	S_{24}	S_{25}	S_{26}
S_{13}	False	True	False
S_{25}	True	False	False
S_{26}	False	False	$W_{26,26}$

Фиг. 5.8. Матрица на предикатите R_3

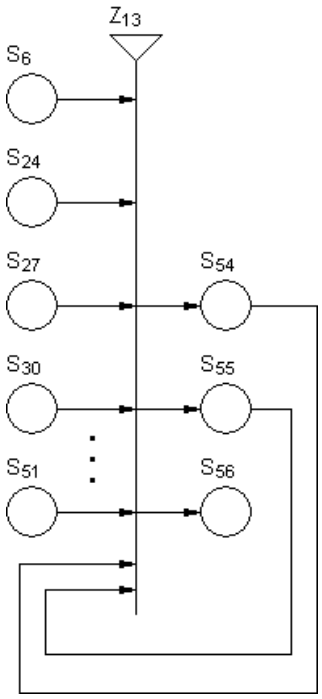
В табл. 5.12 по-долу е представено описание на предиката в абстрактен вид.

Предикат	Описание
$W_{26,26}$	$W_{26,26} = x_2^{\chi}$. Следователно при x_2^{χ}=true и $W_{26,26}$=true

Табл. 5.12. Описание на предикатите в R_3

Преходът Z_3 е обобщеномрежово представяне на скрит неврон, който се активира при постъпване на входно въздействие във формата на ядрото ξ_n', което носи всички входни въздействия от входните неврони на дадената невронна мрежа. Описанието на работата на прехода Z_3 представя и работата и на преходите Z_4 до Z_{12} и Z_{15} до Z_{24}, които извършват еднаква дейност като скрити неврони в конкурентното обучение на двете невронни мрежи.

5.2.4 Описание на преход Z_{13}



Фиг. 5.9. Графично представяне на прехода Z_{13}

Z_{13} = <{S_6, S_{24}, …, S_{51}, S_{54}, S_{55}}, { S_{54}, S_{55}, S_{56}}, R_{13},
∨(∧(S_{24}∨(S_{27}, $S_{31,}$ …, S_{51})), S_6)>

Преходът Z_{13} симулира работата на изходните неврони в невронната мрежа. Целта на този преход е да събере изходните резултати на скритите неврони и да ги обработи, за да получи изходните стойности на всеки един неврон. Позицията S_6 е входна, в нея постъпва ядрото *υ′*. То носи информация за желаните изходни резултати за всяка итерация от обучението на невронната мрежа. На входните позиции S_{24} до S_{51} пристигат ядрата ξ'_1 до ξ'_{10}. Те носят резултатите от всички скрити неврони, които са активирани в текущата невронна мрежа. Това означава, че е възможно да пристигнат от един скрит неврон ξ'_1 до всички скрити неврони ξ'_1,…, ξ'_{10}. На входната позиция S_{54} първоначално не постъпва ядро. Позиция S_{55} приема ядрото *ω′*, което съхранява основни параметри на изходните неврони на невронната мрежа. В табл. 5.13 е показана таблица със структурирано описание на всички ядра, които постъпват в невронната мрежа.

Ядро	Начални характеристики	Позиция на създаване/постъпване	Описание на ядрото
ξ'_1	x_0^{ξ},x_1^{ξ},x_2^{ξ},x_3^{ξ}	S_{24}	Съхранява данни за входните стойности на невронната мрежа
…	…	…	…
ξ'_{10}	x_0^{ξ},x_1^{ξ},x_2^{ξ},x_3^{ξ}	S_{51}	Съхранява данни за входните стойности на невронната мрежа
υ′	x_0^{υ}, x_1^{υ}, x_2^{υ}	S_6	Съхранява данни за желаните изходни стойности на невронната мрежа
ω′	x_0^{ω}	S_{55}	Съхранява изходните резултати на невронната мрежа от всяка епоха

Табл. 5.13. Таблица на ядрата в преход Z_{13}

За правилната работа на невронната мрежа е необходимо постъпилата информация от ядрата да бъде обработена. За целта в този преход служат две характеристични функции. Първата, която се активира е *InitialTranzitionInitialization*. В нея се инициализира първоначално ядрото *ω′*. В него се съхраняват всички онези параметри, които могат да се използват от алгоритмите за обучение на многослойни невронни мрежи. Характеристичната функция се активира при постъпване на ядрото *ω′* на позиция S_{55}. Втората характеристична функция, която се използва в този преход, е *ProcessHiddenNeuronsActivity*. Тя се активира при постъпване на поне едно ядро от типа *LearningInputPattern* (ξ'_1) и едно ядро от типа *LearningOutputPattern* (υ'). Характеристичната функция *ProcessHidden-NeuronsActivity* изчислява изходните резултати на невронната мрежа на

база на постъпилите изходни резултати от скритите неврони. След извършването на тези изчисления ядрата от тип *LearningOutputPattern* и *LearningOutputPattern* преминават към прехода Z_1. В прехода Z_1 ще бъдат извършени изчисленията по текущата итерация или епоха на алгоритъма за обучение на многослойната невронна мрежа. Преходът Z_{13} притежава матрица на преходите R_{13}. В нея има само един предикат, а всички останали стойности са константи. Единственият предикат на прехода е $W_{6,54}$. Той се грижи за едновременното пристигане на ядрата $\xi'_1 ... \xi'_{10}$ и u'. На фиг. 5.10 и табл. 5.16 по-долу са представени матрицата на предикатите R_{13} и формално описание на предиката.

R_{13}	S_{54}	S_{55}	S_{56}
S_6	$W_{6,54}$	False	False
S_{24}	True	False	False
S_{27}	True	False	False
S_{30}	True	False	False
S_{33}	True	False	False
S_{36}	True	False	False
S_{39}	True	False	False
S_{42}	True	False	False
S_{45}	True	False	False
S_{48}	True	False	False
S_{51}	True	False	False
S_{54}	False	False	True
S_{55}	False	True	False

Фиг. 5.10. Матрица на предикатите R_{13}

Предикат	Описание
$W_{6,54}$	Ако е имало ядро в позиция S_{24} на предишната стъпка, то предикатът е „*true*”

Табл. 5.16. Описание на предикатите в R_{13}

Преходът Z_{13} се явява последна стъпка от правия ход на обучение на невронната мрежа. Резултатът от работата на този преход е изчисляване на резултатите на изходните неврони. След приключване на дейността му, в преход Z_1 се активира процесът на корекция на обучителните параметри при завършване на епоха от текущото обучение. Преходът Z_{13} е еднакъв с прехода Z_{25}, поради което последният не е описан отделно.

5.3 Постигнати резултати

Оптимизиране на обучението чрез употреба на оптимизационен алгоритъм

Предложен е оптимизационен алгоритъм по два и повече параметъра. В този случай оптимизацията е направена по степента на обучение и броя на скритите неврони за многослойната невронна мрежа. Причината да се ограничи обобщеномрежовият модел да симулира

работата на две конкурентно обучавани невронни мрежи е породена от софтуера за симулация на обобщени мрежи и подходяща невронна мрежа за оптимизация на три или повече оптимизационни параметъра. До момента не е предлаган подобен оптимизационен алгоритъм за конкурентно обучение на многослойни невронни мрежи.

Ефективно използване на ресурсите на съвременните процесори

Съвременните процесори използват две до четири ядра в един корпус. Този оптимизационен алгоритъм позволява да се използват оптимално наличните изчислителни ресурси на този етап и да се търсят нови алгоритми за паралелно обучение на всички алгоритми за машинно обучение.

Ускорение на процеса на обучение на многослойни невронни мрежи

Ускорението на процеса на обучение на многослойните невронни мрежи се постига чрез конкурентно и независимо обучение на две невронни мрежи, оптимизирайки резултатите по два параметъра. Не се използва паралелна симулация на невронните мрежи като на всеки етап от обучението да се синхронизира работата им. Причината този подход да се избягва е, че уеднаквяването на стъпките за паралелно изпълнение при по-голяма гранулярност на паралелна работа може да доведе до значително забавяне на процеса на обучение при различен брой скрити неврони и параметри на всяка мрежа.

Представяне на нов конкурентен обобщеномрежов модел за обучение на многослойни невронни мрежи

Този модел е първият модел на конкурентно обучавани многослойни невронни мрежи и паралелна оптимизация върху два значими параметъра на оптимизация. Представеното обобщеномрежово моделното описание се базира на публикация [12].

Създаване на конкурентна и статична структура на обобщената мрежа, която позволява лесно увеличаване на броя конкурентно обучавани невронни мрежи

Разглеждайки статичната структура на обобщената мрежа, много лесно може да се забележи нейната симетричност и функционална разделеност на двете работещи невронни мрежи. Причината да се търси подобен подход е възможността да се мултиплицират възможностите за оптимизация на повече невронни мрежи чрез копиране на готови преходи и минимална модификация на характеристичнте функции в началния преход Z_1. Така създаденият обобщеномрежов модел се отличава с възможност за мащабируемост, а въз основа на това и сравнително лесно и бързо увеличаване на броя симулирани невронни мрежи.

Научни и научно-приложни приноси

1. Създаден е обобщеномрежов модел на неврон, използван при мрежи с право разпространение на сигнала, който позволява конкурентна обработка на входните въздействия върху неврона.

2. Създаден е обобщеномрежов модел за паралелна оптимизация на многослоен перцептрон с обратно разпространение на грешката. В разгледания модел оптимизацията е направена по броя на скритите неврони за невронна мрежа с право разпространение на сигнала чрез алгоритъма на златното сечение. Ускорението на процеса на обучение на невронната мрежа с право разпространение на сигнала се постига чрез паралелно обучение на две невронни мрежи.

3. Разработен и симулиран е обобщеномрежов модел за паралелна оптимизация на скрити неврони в невронна мрежа с радиални базисни функции. Предимствата на обобщеномрежовия модел са следните:

- Симулиране на обучение на истинска невронна мрежа и постигане на резултати с точност 4 порядъка след десетичния знак.

- Конкурентно обучение на невронни мрежи;

- За първи път е извършено пресмятането на диференциалните уравнения в обобщена мрежа;

- Намаляване на времето за обучение в зависимост от паралелността на изчислителните ресурси.

4. Създаден е обобщеномрежов модел на алгоритъм за конкурентно обучение на многослойни невронни мрежи. Направена е оптимизация по степента на обучение и броя на скритите невронни за многослойна невронна мрежа. Предимствата на предложеният алгоритъм са:
   - Ефективно използване на ресурсите на съвременните процесори поради паралелността на алгоритъма;
   - Ускорение на процеса на обучение на многослойни невронни мрежи;
   - Създаване на обобщеномрежов модел с цел мащабируемост при скалиране към по-голям брой паралелно симулирани невронни мрежи.

Библиография

[1] Антонов, А., Обзор на обобщеномрежовите модели, описващи функционирането на невронни мрежи, Годишник на секция „Информатика", Съюз на учените в България Том 5, 2012, 120-128.

[2] Атанасов, К. Т., Въведение в теорията на обобщените мрежи, Понтика принт, Бургас, 1992.

[3] Атанасова, В., Изследване на алгоритми за конструиране на обобщеномрежови модели, дисертационен труд по придобиване на

образователната и научна степен "Доктор" в ИИСТ - БАН, София, 2013.
[4] Димитров, Д., Програмен аспект на теорията на обобщените мрежи – оптимизация на алгоритми за изпълнение, оператори за модификация на модели и приложения, дисертационен труд по придобиване на образователната и научна степен "Доктор" във ФМИ – СУ „Св. Климент Охридски”, София, 2013.
[6] Aladjov, H., Generalized Net for Artificial Neural Networks Learning, Proceedings of International Symposium “Bioprocess Systems”, Sofia, 2000.
[8] Alexieva, J., E. Choy, E. Koycheva, Review and bibloigraphy on generalized nets theory and applications. - In: A Survey of Generalized Nets (E. Choy, M. Krawczak, A. Shannon and E. Szmidt, Eds.), Raffles KvB Monograph, No 10, 2007, 207-301.
[9] Anderson, A., J. Silverstein, S. Ritz, J. Randall, Distinctive features, categorical perception, and probability learning: Some applications of a neural model, Psychological Review, Vol 84, No 5, 1977, 413-451.
[11] Antonov, A., Presentation of neuron by generalized net, Issues in the Representation and Processing of Uncertain Imprecise Information: Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets, and Related Topics (K. Atanassov, J. Kacprzyk, M.Krawczak and E. Szmidt, Eds.) Akademicka Oficyna Wydawnictwo EXIT, Warszawa, 2005, 1-10.
[12] Antonov, A., S. Hadjitodorov, Concurrent Algorithm for Learning of Neural Networks, IEEE 6th International Conference ‘Intelligent Systems’, 2012, 225-228.
[13] Antonov, A., Generalized net model for parallel optimization of hidden units in neural networks with radial basis functions, Comptes Rendus de L'Academie Bulgare des Sciences, Vol. 66, No. 9, 2013, 1239-1246.
[14] Arfken, G. B., H. Weber, F. Harris, Mathematical Methods for Physicists, 7th ed.: A Comprehensive Guide, FL: Academic Press, Orlando, 2012, 448-467.
[15] Atanassov, K., Theory of Generalized nets (an algebraic aspect), Advances in Modelling & Simulation, AMSE Press Vol. 1, No. 2, 1984, 27-33.
[19] Atanassov, K., Generalized Nets, World Scientific, Singapore, 1991.
[21] Atanassov, K., S. Sotirov, A. Antonov, Generalized net model for parallel optimization of feed-forward neural network, Advanced Studies in Contemporary Mathematics, Vol.15, No. 1, 2007, 109-119.
[22] Atanassov, K., On Generalized Nets Theory, Prof. Marin Drinov Academic Publishing House, Sofia, 2007.
[23] Atanassov, K., M. Krawczak, S. Sotirov, Generalized Net Model for Parallel Optimization of Feed-Forward Neural Network with Variable Learning Rate Backpropagation Algorithm, Advanced Intelligent system/ IS from theory to practice, Springer, 2010, 361-372.
[25] Bishop, C.M., Neural Networks for Pattern Recognition, Oxford, Oxford University Press, 1995.
[28] Brumfiel, G., High-energy physics: Down the petabyte highway, Nature 469, 19 January 2011, 282-283.
[29] Buck, S., RBF++ A C++ Library for Function Approximation Using Networks of Radial Basis Functions, Munich University of Technology, 2002,

www9.in.tum.de/people/buck/index_e.html
[36] Chu, C., S.K. Kim, Y.A. Lin, Y.Y. Yu, G. Bradski, A.Y. Ng, K. Olukotun, Map-Reduce for Machine Learning on Multicore, Advances In Neural Information Processing Systems 19 – Proceedings of the 2006 Conference, MIT Press, December 2006.
[38] Dean, J., S. Ghemawat, Mapreduce: Simplified data processing on large clusters, Operating Systems Design and Implementation, 2004, 137-149.
[40] Dimitrova, M., Kr. Vasilev, S. Sotirov, Generalized Net Model Of The Process Of The Prognosis Biomass Accumulation With Neural Network, Development in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and related topics. Aplications. Vol. II , System Research Institute, Polish Academy of Science, Warsaw, 2010, 79-90.
[41] Dunlap, R. The Golden Ratio and Fibonacci Numbers, World Scientific, Singapore, 1997.
[45] Faggin, F., VLSI implementation of neural networks, Tutorial Notes, International Joint Conference on Neural Networks, Seattle, WA, 1991.
[46] Finger, S., Origins of neuroscience: a history of explorations into brain function. Oxford University Press US, 48, 2001, ISBN 978-0-19-514694-3.
[47] Flynn, M.J., Some Computer Organizations and Their Effectiveness, IEEE Transactions on Computers, Volume C-21, Issue 9, Sept. 1972, 948 - 960, ISSN 0018-9340.
[58] Gupta, S., What's Machine Learning ? Thanks to GPU Accelerators. You're Already Soaking In It, 2014, http://blogs.nvidia.com/blog/2014/03/25/machine-learning/.
[65] Haykin, S., Neural Networks, Macmillian Publishing Company, New York, 1994.
[67] Haykin, S., Neural Networks and Learning Machines, 3th ed., Englewood Cliffs, NJ: Prentice Hall, 2009.
[69] IEEE CS digital library - http://www.computer.org/portal/web/csdl
[70] IEEE Explorer - http://ieeexplore.ieee.org/Xplore/guesthome.jsp
[71] Jensen, K., Coloured Petri nets and the invariant-method, Theoretical Computer Science, Vol. 14, No. 3, 1981, 317-336.
[72] Kavis, M., Architecting the Cloud: Design Decisions for cloud Computing Service Models (SaaS, PaaS, and IaaS), 2014.
[73] Kawamoto, A., J. Anderson, A neural network model for multistable perceptron, Acta Psychologica, Vol. 59, 1985, 35-65.
[83] Kolev, B., An Algorithm for Transforming a Graph to a Generalized Net, Proceeding of the First International Workshop on Generalized Nets, Sofia, 9th July 2000, 26-28.
[88] Krawczak, M., H. Aladjov, Generalized net model of backpropagation learning algorithm, Proc. of the Third Int. Workshop on Generalized Nets, Sofia, 1 October 2002, 32-36.
[89] Krawczak, M., Generalized net models of multilayer neural networks, Advanced Studies on Contemporary Mathematics, Vol. 7, No.1, 2003, 69-86.
[93] Krawczak, M., On a generalized net model of MLNN simulation, Soft Computing Tools, Techniques and Applications (P. Grzegorzewski, M.

Krawczak, S. Zadrozny, Eds.), Academicka Oficyna Wydawnicza EXIT, Warszawa, 2004, 157-171.
[96] Krawczak, M., A way to aggregate multilayer neural networks, Lecture Notes in Computer Science, Vol. 2, No. 3697, 2005, 19-24.
[97] Krawczak, M. Generalized net models of MLNN learning algorithms, Lecture Notes in Computer Science, Vol. 2, No. 3697, 2005, 25-30.
[100] Krawczak, M., S. Sotirov, K. Atanassov, Multilayer Neural Network Modellig by Generalized Nets, Warsaw School of Information Technologies, 2010.
[101] Krawczak, M., S. Sotirov, E. Sotirova, Generalized Net Model for Parallel Optimization of Multilayer Neural Network with Time Limit, IEEE 6th International Conference 'Intelligent Systems', 2012, 173-177.
[102] LHC Brochure, English version. A presentation of the largest and the most powerful particle accelerator in the world, the Large Hadron Collider (LHC), which started up in 2008. Its role, characteristics, technologies, etc. are explained for the general public., CERN-Brochure-2010-006-Eng., LHC Brochure, English version, CERN, 2010.
[103] LHC Guide, English version. A collection of facts and figures about the Large Hadron Collider (LHC) in the form of questions and answers., CERN-Brochure-2008-001-Eng, LHC Guide, English version, 2008.
[108] McCulloch, W., W. Pitts, A logical calculus of ideas immanent in nervous activity, Bulletin of Mathematical Biophysics, No. 7, 1943, 115-133.
[112] Narendra, K.S., K. Parthasarathy, Identification and control of dynamic systems using neural networks, IEEE Transactions on Neural Networks, Vol. 1, 1990, 4–27.
[113] Natkin, S., Les reseaux de Petri stochastiques et leur application a l'evaluation des systems informatiques, D. diss., Juin 1980.
[116] Nikolova, N., Generalized net representation of backpropagation neural networks, Advances in Modelling & Analysis, D, AMSE Press, Vol. 1, No. 1, 2, 1998, 27-34.
[117] Nutt, G., The formulation and application of evaluation nets, Ph. D. diss., Comp. Sci. Group, Univ. of Washington, Seattle, July 1972.
[124] Radeva, V., M. Krawczak, and E. Choy, Review and bibliography on generalized nets theory and applications. Advanced Studies in Contemporary Mathematics, Vol. 4, 2002, 173-199.
[125] Ramanathan, R., Intel Multi-Core Processors, Leading the Next Digital Revolution, 2005, http://www.intel.com/content/www/us/en/research/intel-labs-multi-core-revolution-paper.html?wapkw=multi+core.
[126] Ramchandani C., Analysis of asynchronous concurrent systems by timed Petri nets, Ph. D. diss, MIT, Cambridge, Mass., Sept. 1973.
[127] Riedmiller, M., H. Braun, A direct adaptive method for faster backpropagation learning: the Rprop algorithm, Proceedings of the ICNN, San Francisco, 1993.
[128] Rumelhart, D., G. Hinton, and R.J. Williams., Learning representations by back-propagating errors, Nature, London, Vol. 323, 1986, 533-536.
[129] Saltzer, J., M. Kaashoek, Principles of Computer System Design : An Introduction, Morgan Kaufmann, 23 Jun 2009, 326.

[130] Shapiro, S., A stochastic Petri nets with applications to modelling occupancy timed for concurrent task systems, Networks Vol. 9, 1979, 375-379.
[132] Shepherd, G.M., C. Koch, Introduction to synaptic circuits, The Synaptic Organization of the Brain, New York: Oxford University Press, 1990, 3-31.
[133] Sotirov, S., Modeling the algorithm backpropagation for learning of neural networks with generalized nets, Part 1 Proc. of the Fourth Int. Workshop on Generalized Nets, Sofia, 23 Sept. 2003, 61-67.
[135] Sotirov, S., Generalized net model of the accelerating backpropagation algorithm, Jangjeon Mathematical Society, 2006, 217-225.
[138] Sotirov, S., M. Krawczak, Modeling the algorithm backpropagation for learning of neural networks with generalized networks -Part 2. In: Atanassov K., Kacprzyk J., M. Krawczak, Szmidt E. (Eds.): Issues in intuitionistic fuzzy sets and generalized nets, Vol. 3, Warsaw School of Information Technology, Warsaw, 2006, 65-69.
[143] Sotirov, S., D. Orozova, E. Sotirova, Generalized net model of the process of the prognosis with feedforward neural network, Proc. of the XVI-th International Symposium on Electrical Apparatus and Technologies, SIELA, Vol.1, 2009, 272-278.
[145] Sotirov, S., E. Sotirova, M. Krawczak, Application of Data mining in Digital University: Multilayer perceptron for Lecturer's Evaluation with Intuitionistic Fuzzy Estimations, Issue on IFS and GNs, Vol. 8, Warszawa, 2010, 102-108.
[146] Sotirov, S., Generalized net model of the time delay neural network, Issue on IFS and GNs, Vol. 9, Warszawa, 2010, 125-131.
[147] Sotirov, S., M. Krawczak, K. Atanassov, Generalized Net Model for Parallel Optimization of Multilayer Perceptron with Momentum Backpropagation Algorithm, 5th International IEEE Conference "Intelligent Systems", London, 2010, 281-285.
[152] Sotirov, S., M. Krawczak, Modelling Layered Digital Dynamic Network by a Generalized Net, Issue on IFS and GNs, Vol. 9, Warszawa, 2011, 84-91.
[153] Sotirov, S., Generalized Net Model of Iris Recognition Using Neural Networks, Annual of "Informatics" Section Union of Scientists in Bulgaria, Vol. 4, 2011, 127-133.
[154] Sotirov, S., M. Krawczak, Generalized Net Model for Parallel Optimization of Multilayer Perceptron with Conjugate gradient Backpropagation Algorithm, Development in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and related topics. Aplications, Vol. II, System Research Institute, Polish Academy of Science, Warsaw, 2012.
[158] Sutter, H., The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software, Dr. Dobb's Journal, 2005, http://www.gotw.ca/publications/concurrency-ddj.htm.
[159] Sutter, H., Understanding Parallel Performance, Dr. Dobb's Journal, 2008, http://www.ddj.com/cpp/211800538.
[160] Trifonov, T., K. Georgiev, GNTicker - A software tool for efficient interpretation of generalized net models, Issues in Intuitionistic Fuzzy Sets and Generalized Nets, Warszawa, 2004.