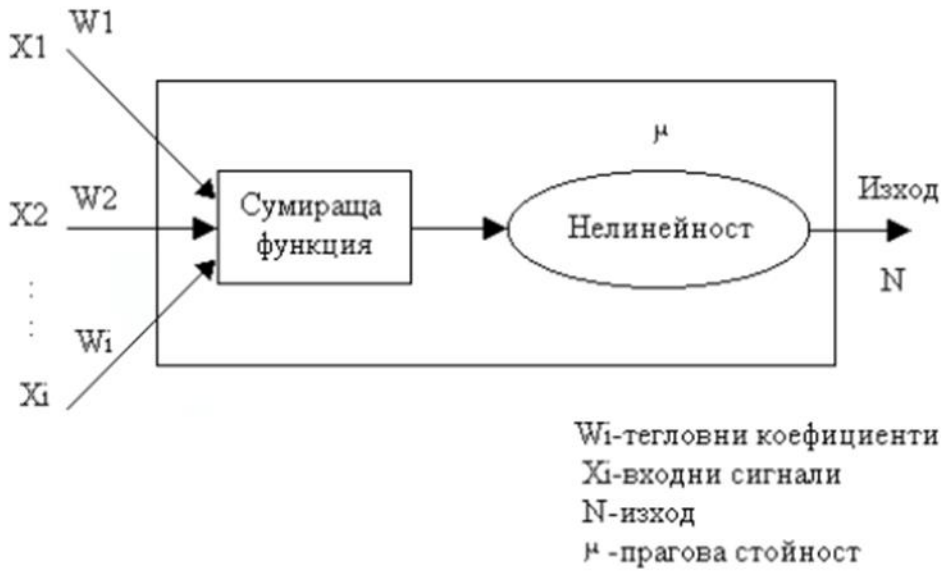
Тема 1/Занятие 7/Аудиторно упражнение

Обучение на невронни мрежи.

Неврон на McCulloch-Pitt

Невронът на McCulloch-Pitt, показан на фигура 3, е първият изчислителен модел на неврон. Предложен е през 1943 г. от Уорън МуКълок (невролог) и Уолтър Питс (математик). Чрез него се илюстрира идеята на използването на невронните мрежи за различни изчисления. Моделът се счита за универсален тъй като всяка логическа функция може да бъда изчислена с мрежа от неврони на McCulloch-Pitt. Същевременно всяка крайна последователност от дискретни действия може да бъде симулирана с рекурентни невронни мрежи от такива неврони. Ето защо те представляват основа за създаване на други процесорни елементи чрез използване на различни нелинейни функции.



Фиг. 3. Неврон на McCulloch-Pitt

Както показва фиг. 3, на входа на неврона постъпват входните сигнали, X=**{x_1, x_2, ….., x_i},** които както и изходния сигнал **Yса** двоични т.е. $x_i \in \{0,\ 1\}$ и $y \in \{0,\ 1\}$.

Входовете биват два вида – **възбудителни** и **забранителни**. Теглата, свързани със възбудителните входове $x_1, x_2, ..., x_k$ са отрицателни. Всеки ненулев входен сигнал на някой от забранителните входове генерира нулев изходен сигнал. Те се обобщават от сумираща функция и постъпват на входа на блок, който реализира нелинейна функция. Нелинейната функция трансформира входа от сумиращата функция в изход, който определя изходния сигнал на неврона. Нелинейната функция има определена прагова стойност (параметърът μ на фиг. 3). Когато сумата от произведенията на входовете $\mathbf{X_i}$ с теглата $\mathbf{W_i}$ надхвърли този праг, невронът се активира (извежда 1), а в противен случай остава в покой (извежда 0).

$$y = \begin{cases} y{=}1,\ \textit{когато}\ \sum_{j=1}^{k} w_j^+ x_j \geq \theta\ \textit{или}\ x_i{=}0,\ i{=}k{+}1,...,m \\ 0,\ \textit{когато}\ \sum_{j=1}^{k} w_j^+ x_j < \theta <\ \textit{или}\ \ i{=}k{+}1,...,m,\ \textit{такова че}\ x_i{=}1 \end{cases} \qquad (3)$$

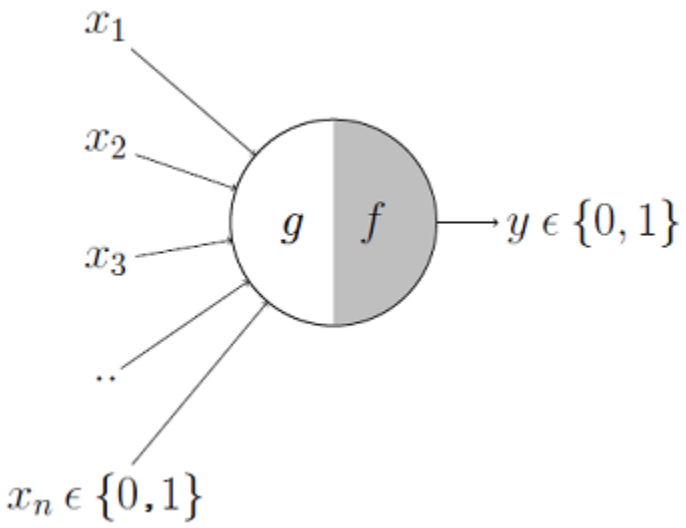
За правилната работа на неврона трябва да е изпълнено условието

$$\sum_{j=1}^{k} w_j^+ - w_i^- < \theta, \quad \forall i = k+1,...,m \qquad (4)$$

При невронът на McCulloch-Pitt теглата са фиксирани, т.е. не се настройват с обучение. [5]

Неговата работа може да бъде описана със следващите прости примери.

Нека да разглеждаме неврона условно като съставен от две части, както е показано на фиг. 4. Лявата част *g* приема входните сигнали, сумира ги и предава резултата на дясната част *f*, която взема решението.



Фиг. 4. Опростен модел на неврон на McCulloch-Pitt

Например**:** Нека да неврона да вземе решение *дали да бъде гледан или не футболен мач по телевизията.*

Всички входни и изходни променливи са булеви, т.е. {0: Ще го гледам, 1: Няма да го гледам}.

Нека:

- ***x_1*** има значение *Играе най-любимия ми отбор*
- ***x_2*** има значение *Играе отбор, който харесвам*
- ***x_3*** има значение *Не съм вкъщи*
- ***x_4*** има значение *Играе отбор, които много харесвам*

В този пример входа ***x_3*** е забранителен. Ако *x_3 е 1 (не съм вкъщи*) изходната функция ще има стойност 0 независимо от стойността на сигнала на останалите входове.

Входовете ***x_1, x_2 и x_4*** са възбудителни, но нито един от тях не може самостоятелно да задейства неврона. Това се случва само, когато сумата им надвиши определена прагова стойност, както е описано в израз (3).

Реализиране на булеви функции с помощта на неврон на McCulloch-Pitt

M-P неврона е подходящ да се използва за реализация на булеви функции тъй като входовете и изходите му са булеви.

В следващите примери да приемем, че неврона има два входа и в зависимост от това ли сумата от техните стойности е надвишава или не определена прагова стойност, неврона ще генерира или не изходна функция.

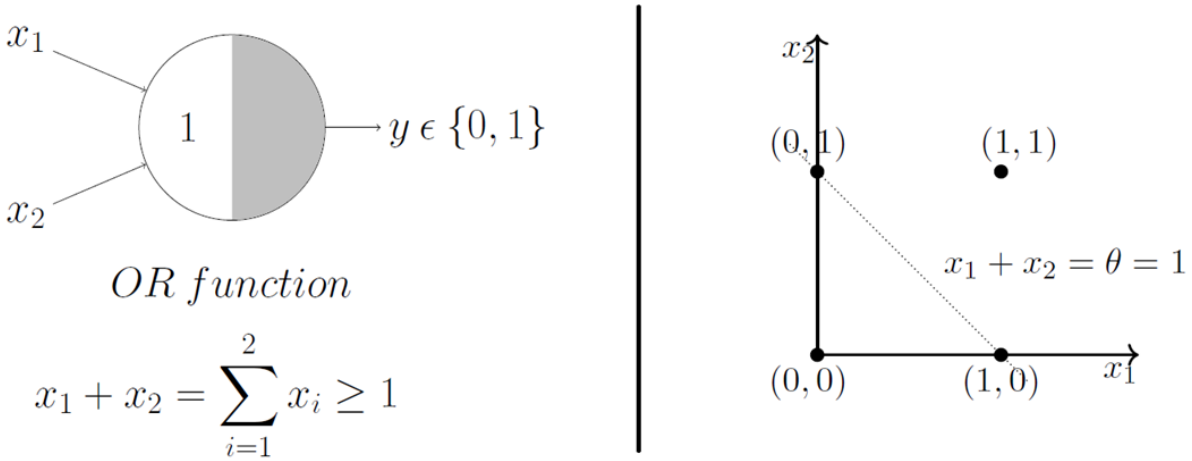
Реализация на логическа функция ИЛИ (OR)

Реализирането на булевата функция ИЛИ от неврона (фигура 5) е равнозначно на генерирането на изходен сигнал ***y = 1*** когато на поне един от входовете има подадена логическа 1 (за справка вж. таблица 1) т.е., ***g(x) ≥ 1***.

Табл. 1. Таблица на истинност на функция ИЛИ за две входни променливи.

x_1	x_2	ИЛИ
1	1	1
1	0	1
0	1	1
0	0	0

Фигура 5 представя четирите възможни комбинации от стойностите на булевите входните променливи $\mathbf{x_1}$ и $\mathbf{x_2}$ като координати на точки в равнина. Уравнението: $x_1 + x_2 \geq 1$ е агрегиращото уравнение на неврона и показва границата на решението. Всички комбинации от стойности на входните променливи, които са координати на точки, лежащи **ПОД линията** ще изведат решение **0**, а тези които лежат **НА** или **НАД** нея ще изведат **1** като резултат.



Фиг. 5 Реализация на логическа функция ИЛИ

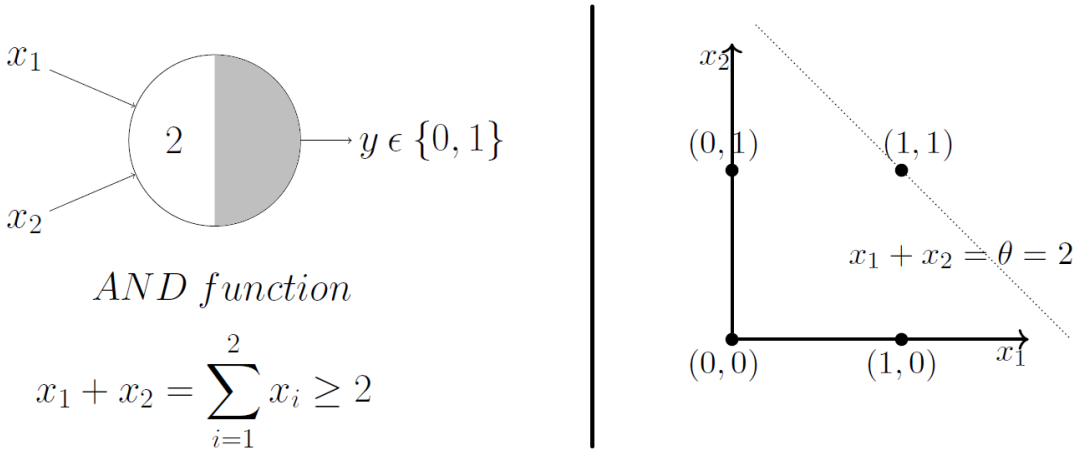
По този начин М-Р невронът е обучен на линейна граница на решение. Той разделя входните набори на два класа - положителни и отрицателни. Положителните (които извеждат 1) са тези, които се намират **ВЪРХУ** или **НАД** границата на вземане на решение, а отрицателните (които извеждат 0) са тези, които се намират ПОД границата на вземане на решение.

Реализация на логическа функция И (AND)

Реализирането на булевата функция И от неврона е равнозначно на генерирането на изходен сигнал $y = 1$ само когато има подадена логическа 1 и на двата входа едновременно (за справка вж. таблица 2) т.е., $g(x) = 2$.

Табл. 2. Таблица на истинност на функция И за две входни променливи.

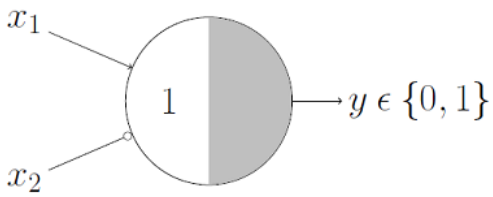
x_1	x_2	**AND**
1	1	1
1	0	0
0	1	0
0	0	0



Фиг. 5 Реализация на логическа функция И

В този случай агрегиращото уравнение на неврона, което показва границата на решението е $x_1 + x_2 = 2$. Всички точки лежащи **ВЪРХУ** или **НАД** линията – в случая точката с координати (1,1), извеждат 1.

Реализация на функция с потискащ вход



$x_1\ AND\ !x_2^*$

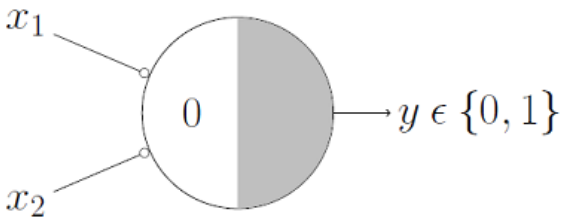
Фиг. 6. Реализация на логическа функция ИЛИ

Входът x_2 е потискащ, така че винаги когато $x_2 = 1$, изходът ще бъде 0. В резултат изходния сигнал ще бъде равен на 1 само когато $x_1=1$ и $x_2=0$, (табл. 2). Очевидно е, че праговият параметър трябва да бъде 1.

Таблица 2. Таблица на истинност на логическа функция ??????

x_1	x_2	$x_1\ AND!\ x_2$
1	1	0
1	0	1
0	1	0
0	0	0

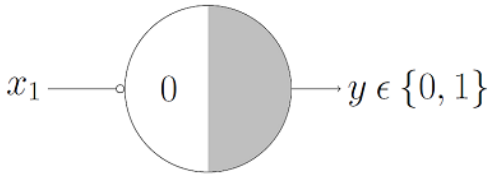
Реализация на логическа функция ИЛИ - НЕ (NOR)



Фиг. 7. Реализация на логическа функция ИЛИ-НЕ

За да се генерира на изхода сигнал ***y*** = 1е необходимо на всички входове да се подаде 0 и праговия параметър да се определи като 0.

Реализация на логическа функция НЕ (NOT)



Фиг. 8. Реализация на логическа функция НЕ

Този неврон извежда 0 когато на входа му е подадена 1 и извежда 1, когато на входа му е подадена 1. За да се постигне такова „поведение“ на неврона входа му трябва да се определи като инхибиращ вход и праговия параметър да се зададе като 0.

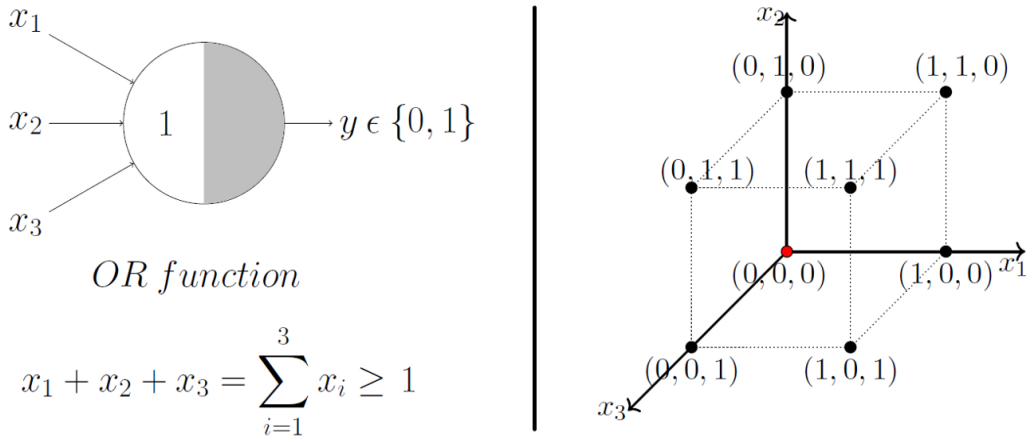
Реализация на логическа функция ИЛИ (OR) с три входни променливи

Реализирането на булевата функция ИЛИ с три входни променливи от неврона (фигура 9) е равнозначно на генерирането на изходен сигнал ***y = 1*** когато на поне един от входовете има подадена логическа 1 т.е., ***g(x) ≥ 1***.

Табл. 3. Таблица на истинност на функция ИЛИ за три входни променливи.

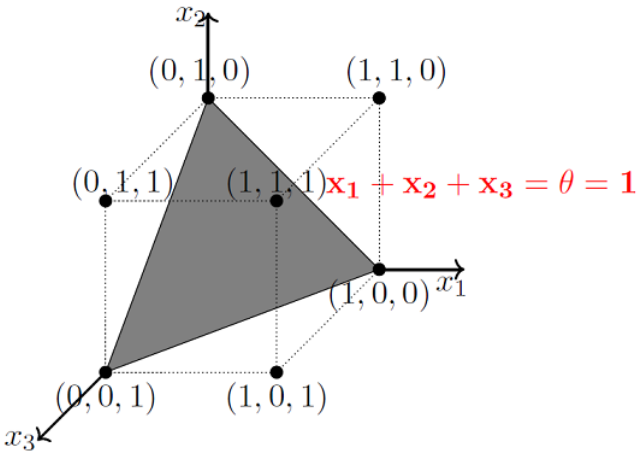
x_1	x_2	x_3	**ИЛИ**
1	1	1	1
1	1	0	1
1	0	1	1

1	0	0	1
0	1	1	1
0	1	0	1
0	0	1	1
0	0	0	0



Фиг. 9. Реализация на логическа функция ИЛИ на три променливи

Както показва таблица 3, възможни са 8 комбинации от стойности на входните променливи. На фигура 9 е показана границата на решение в триизмерното пространство.



Фиг. 10. Граница на решение на логическа функция ИЛИ на три променливи

Всички състояния на входните променливи, определящи точки в пространството, които лежат **ВЪРХУ** или **НАД** равнината, определена от уравнението: $x_1 + x_2 + x_3 = 1$ (положителното полупространство), ще доведат до изход 1, и всички които лежат **ПОД** тази равнина (отрицателно полупространство) ще доведе до изход 0.

Анализът на представените примери води до извода, че правилното определяне на праговия параметър на М-Р неврона позволява реализацията на линейно разделими булеви функции.

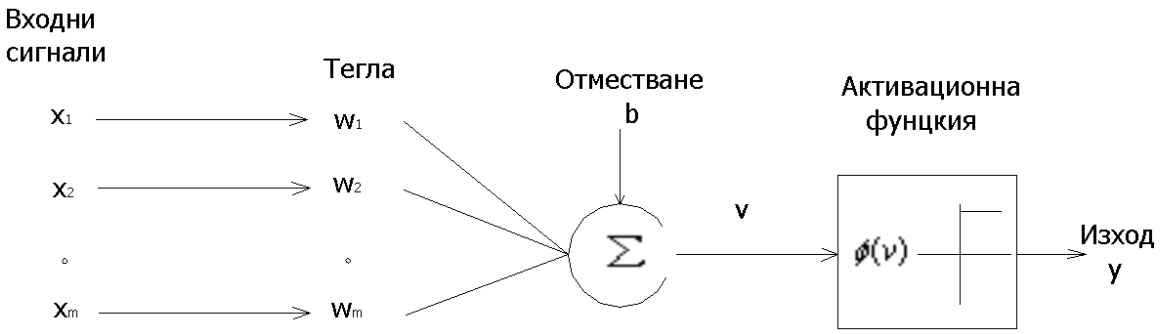
М-Р неврона притежава редица недостатъци:

- Не се разпознават входни променливи, които не са бинарни;
- Определянето на стойността на праговия параметър не може да стане автоматично;
- Не е възможно да се определят различни приоритети на различните входове;
- Не е възможно да се реализират функции, които не са линейно отделими, например функцията XOR.

Перцептрон

Описаните ограничения са преодолени в предложеният през 1957 г. от Frank Rosenblatt обобщен изчислителен модел на неврон при който теглата и праговете могат да бъдат научени с течение на времето.

Перцептронът, представен е на фиг. 4, представлява еднослойна невронна мрежа с един елемент, който често е негов сумиращ процесор. Може да работи както с двоични така и с аналогови входни величини. Използва нелинейна активационна функция (прагова, сигмоидна и др.)



Фиг. 10.

При него изходът е явна функция на входа. Това е типично за всички прави мрежи. Изходът на суматора се изчислява като

$$v = \sum_{j=1}^{m} w_j x_j + b = \sum_{j=0}^{m} w_j x_j, \tag{5}$$

където $w_0 = b$, а x_0 е сигнал с постоянна стойност 1.

Изходният сигнал y на перцептрона се определя с формулата :

$$y = \phi(v) = \begin{cases} 1, & ako \quad v \geq \theta \\ 0, & ako \quad v < \theta \end{cases} \tag{6}$$

където θ е праг на активация на перцептрона, и ако стойността му не е спомената специално се счита, че е нула $\theta = 0$.

При перцептрона се използва обучение с учител. При този тип обучение критерият за обучението се задава с комплект примерни входно-изходни последователности за желаната работа на мрежата от типа

$$\{p_1, d_1\}, \{p_2, d_2\}, ..., \{p_Q, d_Q\}, \qquad (7)$$

където p_Q е образ, подаван на входа на мрежата, а d_Q е съответната правилна (желана) стойност на изхода на мрежата при подаване на p_Q. С прилагане на входните елементи към мрежата се сравняват получената стойност на изхода на мрежата с желаната стойност, зададена от съответното d. След това се стартира алгоритъм за обучение, посредством който се променят (настройват) теглата и отместванията на мрежата така, че получаваната стойност на изходите на мрежата да се доближават до желаната.

Алгоритъмът, прилаган при обучението на перцептрона е следният :

ако имаме d=0 и $f(w^T x) = 1$, то тогава $w_{\_new} = w_{\_old} - x$,

ако имаме d=1 и $f(w^T x) = 0$, то тогава $w_{\_new} = w_{\_old} + x$,

ако имаме d= $f(w^T x)$, то тогава $w_{\_new} = w_{\_old}$,

където с $w_{\_new}$ се означава новата, а с $w_{\_old}$ старата стойност на тегловия коефициент от преходната интерация. С x се означава съответната стойност на p_Q, подадена на входа, а с d- съответното d_Q. Функцията $f(w^T x)$, имаща релеен характер, приема в случая стойности 0 или 1, т.е.

$$y = f(v) = \begin{cases} 1, & v \geq \theta \\ 0, & v < \theta \end{cases} \qquad (8)$$

Ако бъде въведена грешка $e = d - f(w^T x)$, правилото за обучение може да бъде представено в следния компактен вид : $w_{\_new} = w_{\_old} + ex$.

Горното правило може да се обобщи за един слой от k на брой перцептрони, използващи едни и същи входни сигнали $x = [x_1\ x_2\ \ldots\ x_m]^T$. В този случай формулите за настройка на вектора с теглата w_i и отместване b_i, свързани с i-тия перцептрон, са равни на : $w_{i\_new} = w_{i\_old} + e_i x, \quad b_i = b_i + e_i$,

където $e_i = d_{i.} - f(w_i^T x + b_i)$.

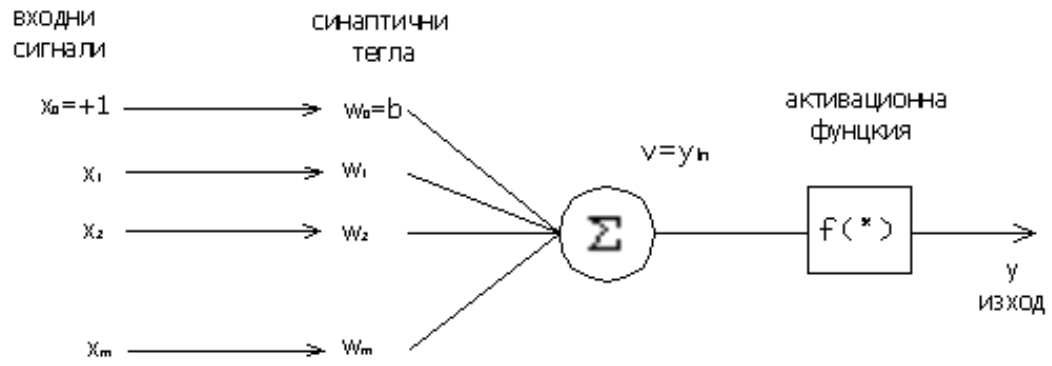
Мрежите на Hopfield

Мрежите на Hopfield обикновенно се състоят от едно ниво, което приема на вход 1 или 0. Тегловните коефициенти са цели числа. Всички неврони са напълно взаимносвързани и всеки от тях се явява като входен, изходен и междинен. Изходът на всеки неврон внася сигнал на входа му.

Направени са подробни изследвания и някои разширения на мрежата на Hopfield. Установено е, че тя може да се използва за оптимизационни задачи (например задачата за търговския пътник), реконструкция на образ от зашумени данни и други. При оптимизационните задачи се търси конфигурация на мрежата, която да минимизира т.нар. енергийна функция. **Енергийната функция е функция на състоянието, която винаги намалява(или остава непроменена)при еволюция на мрежата съгласно промяната в активността на невроните.** Тази функция има локален минимум[1] при стабилно състояние на мрежата. Приложима е, ако тегловните коефициенти в мрежата са симетрични,т.е. силите на връзките са симетрични - $w_{ij}=w_{ji}$.

Адаптивен линеен елемент - ADAptive LINear Element (ADALINE)

Адаптивният линеен елемент - ADAptive LINear Element (ADALINE) е предложен през 1962 г. от Bernard Widrow. Той представлява единичен неврон (фиг.3.2) с линейна активационна функция и настройващи се тегла. Алгоритъмът на Windrow-Hoff за настройка на теглата се базира на метода на най-малките квадрати и е пример за използване на обучение при настройката. Той е в основа за различни видове адаптивна филтрация използвана при обработка на сигнали.



Фиг. 3.2

За обучението на мрежите от тип ADALINE е необходимо да се разпалага с обучаваща извадка, представляваща набори от входни сигнали и желан (еталонен) изход на мрежата за всяки от тях. При подаване на всеки *k*-ти комплект входни сигнали, се определя изходът от невронната мрежа и грешката e_k между него и еталонния изход при зададената комбинация на входа . Целта на обучението е да бъдат настроени теглата $w = [w_1\ w_2\ \ldots\ w_m]^T$ така, че да бъде минимизирана интегралната грешка $\sum_{i=1}^{p} e_i$ за цялата обучаваща съвкупност. В различни варианти на обучаващите алгоритми промяната на теглата се прави след всеки компонент от обучаващата серия или след едно цялостно нейно изчерпване (т.е. след всяка епоха). Във всички случаи

промяната на теглата е пропорционална на съответния входен сигнал, грешката и параметър η , наречен скорост на обучение.

Мрежите Адалайн принадлежат към общия клас алгоритми, наречени адаптивни линейни филтри. Те намират приложение в много области като например за проектиране на изравняващи филтри при високоскоростни модеми, за адаптивни ехокомпенсатори при телефонни разговори на големи разстояния и сателитни комуникации, както и за предсказване на сигнали. Намират приложение и в медицината например за подтискане на шума от биене на сърцето на майката при ембрионална електрокардиография. [3]

Литература:

1. David Kriesel, A Brief Introduction to Neural Networks, достъпно на http://www.dkriesel.com/en/science/neural_networks, посетено на 12.08.2022 г.

2. Терехов В. А., Ефимов Д. В., Тюкин И. Ю. Нейросетевые системы управления. — М.: Высшая школа, 2002. — 184 с. — ISBN 5-06-004094-1.

3. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика = Neural Computing. Theory and Practice. — М.: Мир, 1992. — 240 с. — ISBN 5-03-002115-9.

4. Хайкин С. Нейронные сети: полный курс = Neural Networks: A Comprehensive Foundation. 2-е изд. — М.: Вильямс, 2006. — 1104 с. — ISBN 0-13-273350-1.

5. McCulloch-Pitts Neuron — Mankind’s First Mathematical Model Of A Biological Neuron | by Akshay L Chandra | Towards Data Science