ОБУЧЕНИЕ ЧРЕЗ НАСТРОЙВАНЕ НА ПЕРЦЕПТРОН

Лекция 14

- **Перцептроните** могат да се разглеждат като ***специален вид невронни мрежи***.
- **Перцептронът е най-простата възможна невронна мрежа, *защото има само един подобен на неврон елемент*.**
- .

ПРЕДСТАВЯНЕ НА ПЕРЦЕПРОН

Това е невронна мрежа, в която:

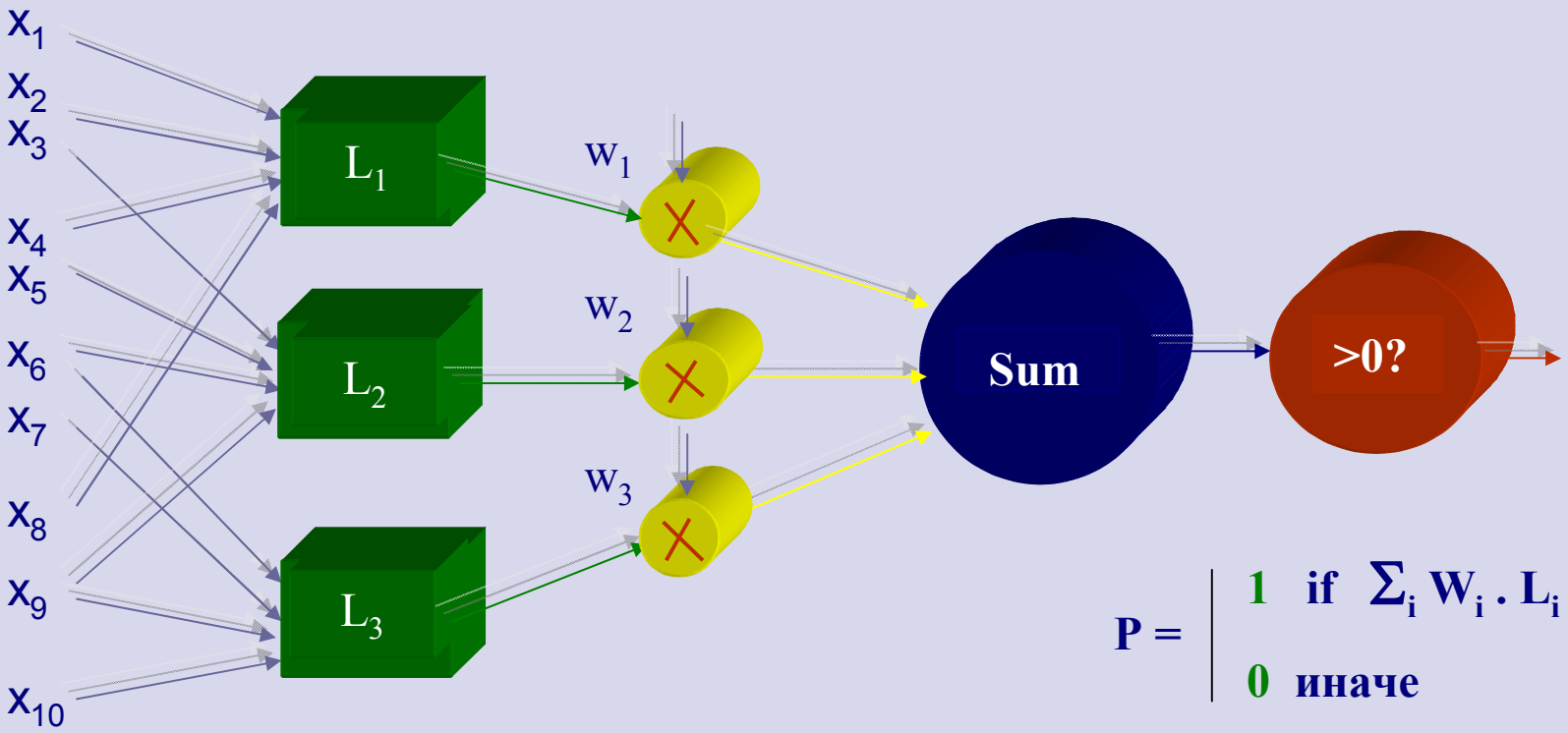
Има само един неврон;

Входовете са двоични: позволени са само 0 и 1;

Логическите кутии могат да бъдат между входовете на перцептрона и теглата на перцептрона. Всяка логическа кутия може да се покаже като таблица, която създава изходна стойност от 0 или 1 за всяка комбинация от 0 или 1-ци, които могат да се появят в нейните входове;

Изходът на перцептрона е 0 или 1 в зависимост от това дали сумата от теглата на изходите на логическите кутии е по-голям от прагът. Фиг. 1;

Фиг. 1 Представяне на перцептрон P



$$P = \begin{cases} 1 & \text{if } \sum_i W_i \cdot L_i > T \\ 0 & \text{иначе} \end{cases}$$

Ако сумата от теглата на изходите на логическите кутии е по-голяма от 0 изходът на перцептрона е 1 и перцептронът иска да каже **ДА**, класът е разпознат. В противен случай перцептронът казва **НЕ** т.е. класът не е разпознат.

Допускаме, че **изходът** на **i**-тия логически блок е $\mathbf{l_i}$, по-нататък допускаме, че i-тото **тегло** е $\mathbf{w_i}$ и накрая допускаме, че прагът е **T**. Така изходът на входния перцептрон P е даден със g горната формула:

- **Логическите кутии виждат само част от входовете на перцептрона**.
  - Всички комбинации на входните 0-ли и 1-ци за една логическа кутия се записват във **входна таблица** с подходящ изход 0 или 1.
  - Броят на редовете в таблицата е **2n** ако имаме **n** входа и всички възможни комбинации от **0-ли** или **1-ци** се поддържат. Тази експоненциална зависимост между броя на входовете и броя на комбинациите от 0 и 1 става непрактично за логическите кутии на перцептрон с много входове. *Затова е измислено всеки блок на перцептрона да следи само малък брой входове*.
- **Перцептронът може да е диаметърно ограничен**.
- **При преките перцептрони** *всяка логическа кутия има само един вход и изходът е винаги същия както този вход*. Той може да бъде представен като перцептрон без логически кутии.

ПРОЦЕДУРАТА ЗА КОНВЕРГЕНЦИЯ НА ПЕРЦЕПТРОН ГАРАНТИРА УСПЕХ КОГАТО Е ВЪЗМОЖЕН УСПЕХ

- Има процедура, *която открива сполучливо множество от тегла* за перцептрон, ако такова сполучливо множество от тегла съществува.
- Започва се с всички тегла от инициализиращото множество с 0. После перцептронът се пробва с всички примери по един в даден момент. Когато перцептронът сгреши се променя теглото така, че да се направи грешката по-малко вероятна, иначе не се прави нищо.
- Може да се направи по-голяма вероятността да се получи ДА следващия път ако се увеличат примерно с 1 теглата свързани към тези логически блокове, които създават 1-ци в момент когато е направена грешка. Съответно когато перцептронът казва ДА когато трябва да каже НЕ трябва да намалите теглата свързани към логическите блокове, които създават 1-ци с намаление равно на 1.
- Изходите на логическите кутии и теглата могат да се представят в термините на вектори като **$(l_1, l_2, \ldots\ldots, l_n)$ и $(w_1, w_2, \ldots\ldots, w_n)$.**

Обучение на перцептрон

- Докато перцептронът даде правилен резултат за всеки обучаващ пример:
  - Ако перцептронът даде грешен отговор:
    - Ако перцептронът даде НЕ когато трябва да каже ДА добави логическа кутия и изходен вектор към вектора на теглото.
    - Иначе извади вектора изход на логически блок от вектора на теглата.
  - Иначе нищо.

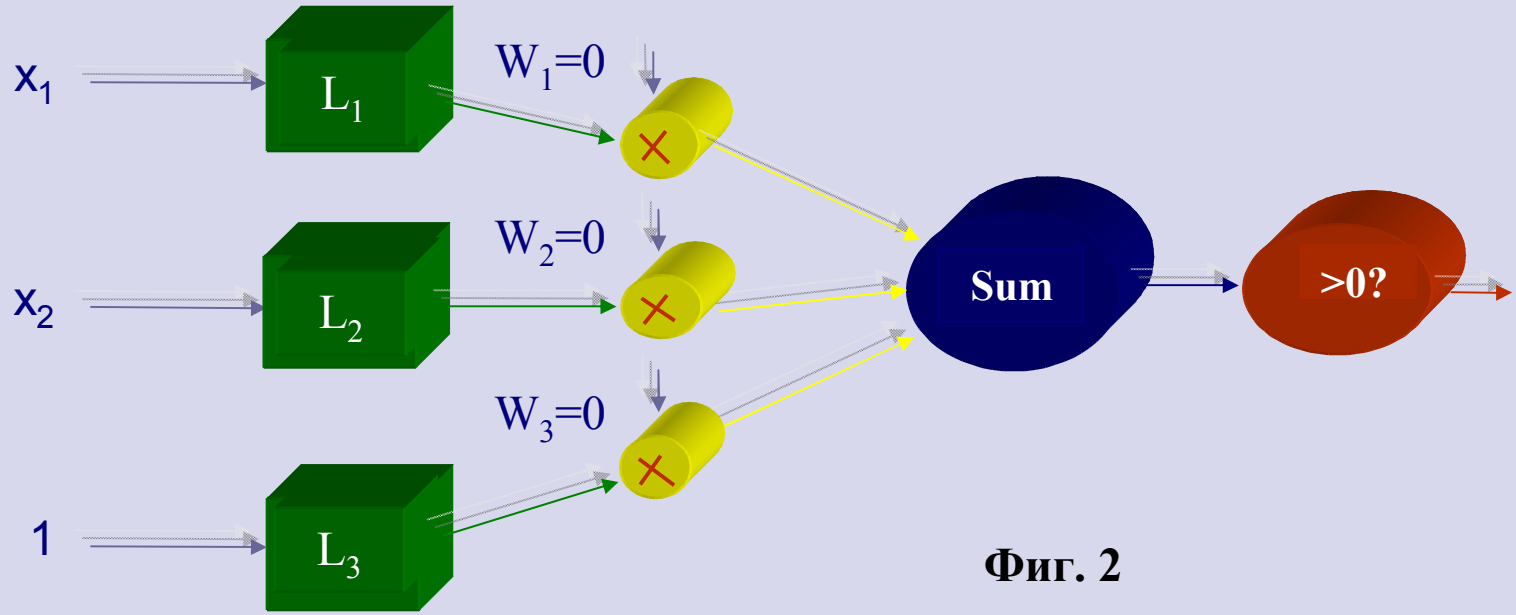
Пример:

- Разглежда се напълно пряк перцептрон / фиг. 2 /, който трябва да се научи да изпълнява функцията логическо ИЛИ като се започва от произволни стойности. Тъй като перцептронът е **напълно пряк изходният вектор** на логическата кутия **(L_1,L_2,L_3)** е същия като **входния вектор (X_1,X_2,X_3) .** Така входните примери и съответните изходи на логическите кутии са както следва за логическо ИЛИ:

Пример:	$X_1 = (l_1)$	$X_2 = (l_2)$	$X_3 = (l_3) = 1$	Желан изход
1	0	0	1	0
2	0	1	1	1
3	1	0	1	1
4	1	1	1	1

• Даденият перцептрон е напълно пряк с два входа и един изход, който отговаря на прага. Той е готов да се обучава да разпознава логическо ИЛИ .

• Обхождайки тези примери перцептронът евентуално ще научи логическото ИЛИ след четири промени.

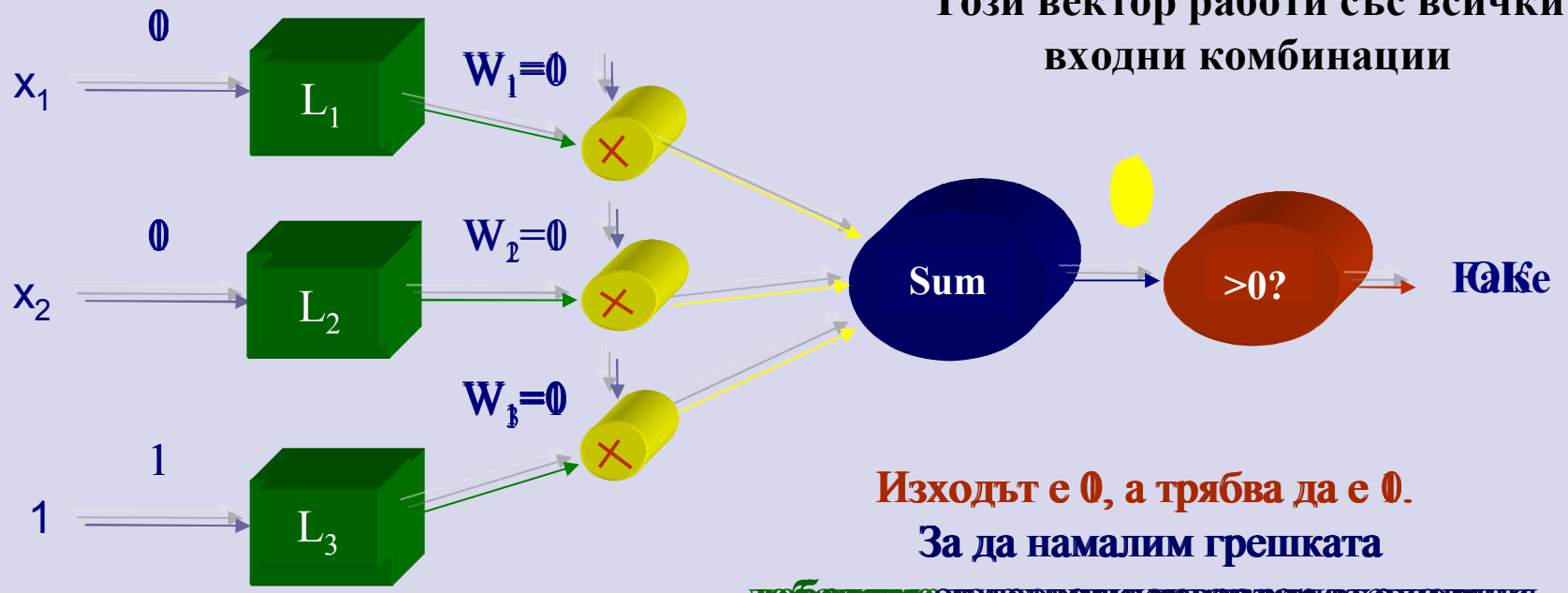


Фиг. 2

Първата промяна става след грешката при втория пример през първия поглед. В този момент векторът на теглата е **(0,0,0)** така че изходът е нула, а той трябва да бъде 1. Съответно когато *входния вектор* (0,1,1) **се добави** към *вектора на теглата,* **новия вектор на теглата ще бъде (0,1,1).** Следващите два примера през първото минаване връщат 1-ци както трябва така че не са необходими по-нататъшни промени през първото минаване.

През второто минаване първият пример *връща 1, но трябва да получим 0*. Съответно входният вектор (0,0,1) **е изваден** от тегловния (0,1,1) така се създава **нов тегловен вектор (0,1,0).** С тази промяна обаче третия пример дава грешка (*връща 0 когато трябва да даде 1).* Съответно входния вектор (1,0,1) се добавя към тегловния вектор (0,1,0) и получаваме **нов тегловен вектор (1,1,1)** .

После първият пример отново дава грешка през третия пас. Резултатът *трябва да е 0 а той е 1*. **Изваждаме** входния вектор (0,0,1) от тегловния вектор (1,1,1) . **Остава нов тегловен вектор (1,1,0), който работи с всички примери**.



Лог. “И”

Този вектор работи със всички входни комбинации

Изходът е 0, а трябва да е 0.

За да намалим грешката

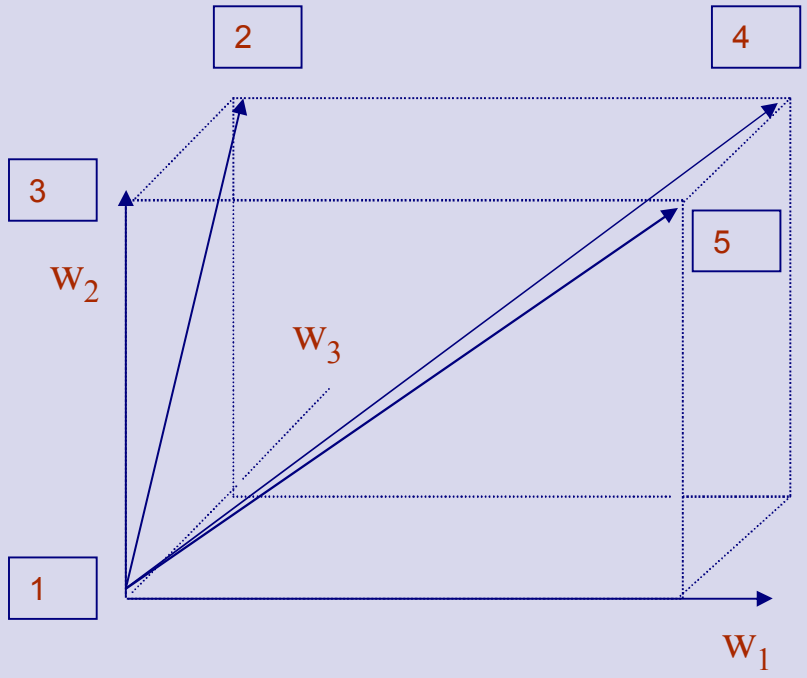
добавяме към тегловния вектор входния.

Новият тегловен вектор е (010)

Пример:	$X_1 = (I_1)$	$X_2 = (I_2)$	$X_3 = (I_3) = 1$	Желан изход
1	0	0	1	0
2	0	1	1	1
3	1	0	1	1
4	1	1	1	1

- ***Забележка***: Винаги входна комбинация от информационни входове и един вход 1 и съответните тегла дават ненулев праг при тези добавяния и изваждания. Не може да се получи (0 0 0) плюс (0 0 0).

- Тъй като имаме три тегла - две за входове и едно за праг - всяка тегловна комбинация може да се покаже като три-измерен тегловен вектор както е показано на фиг.3 като по време на обучението тяхната дължина нараства или се свива.



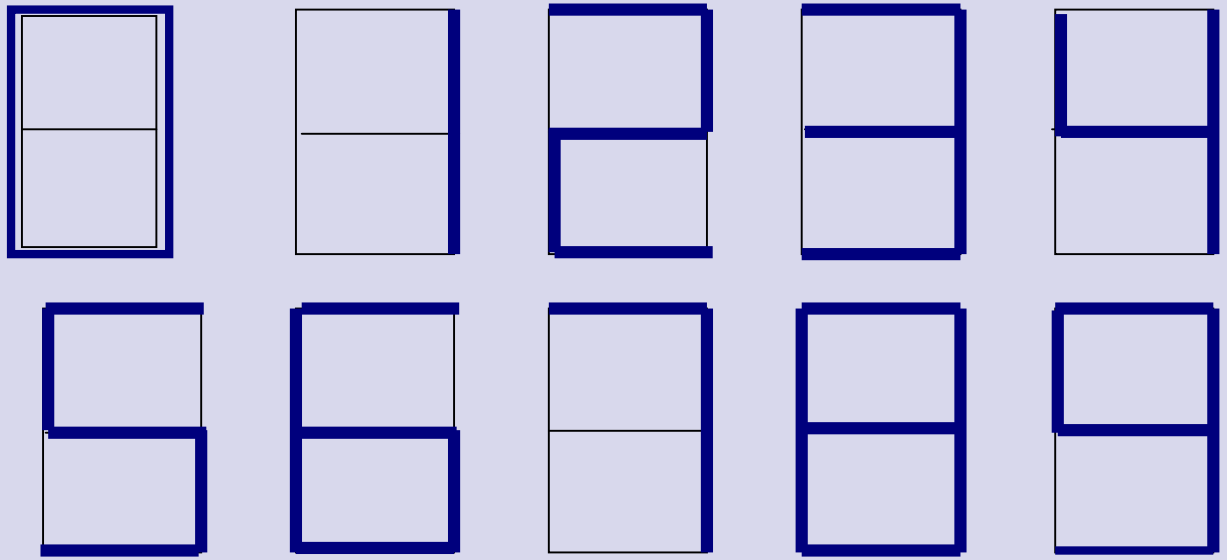
Фиг. 3

Какво може и какво не може перцептрона

Процедурата за **приближаване (сходимост)** (конвергенция) на перцептрона е елегантна. Лесно е да запали ентусиастите относно тяхната голяма мощност. От друга страна обаче ще бъде изяснено, че перцептроните не могат да решат някои задачи.

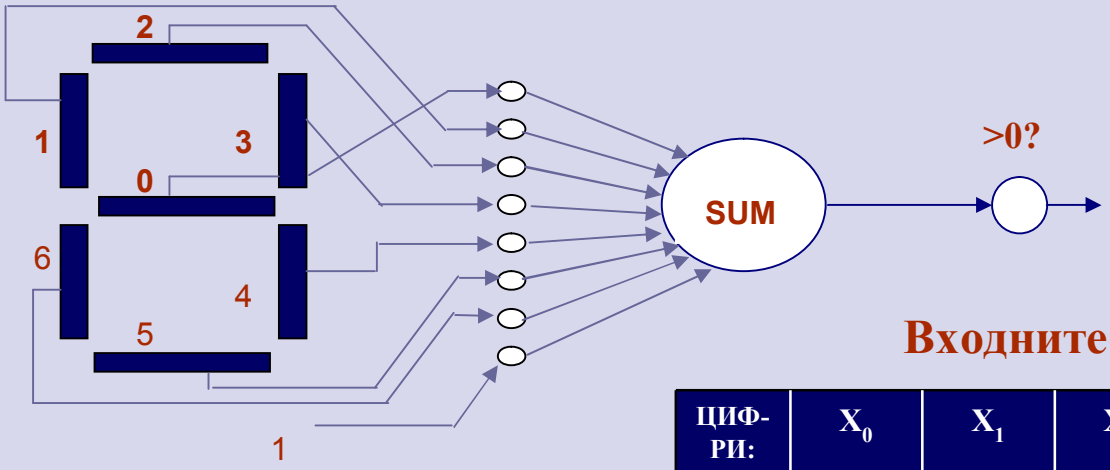
Напълно прекият перцептрон може да се научи да идентифицира цифрите.

Разглеждаме проблемът за разпознаване на цифри, които са направени от превключване на подходяща комбинация от седем възможни сегменти фиг. 4.



Фиг. 4

На фиг. 5 е даден перцептрон, който разпознава цифри. Седем от входовете са свързани към седем линейни сегмента. Осмият винаги е свързан към вход – 1, който определя прага.



фиг. 5

Входните примери са както следва:

ЦИФ-РИ:	X_0	X_1	X_2	X_3	X_4	X_5	X_6
0	0	1	1	1	1	1	1
9	1	1	1	1	1	1	0
8	1	1	1	1	1	1	1
7	0	0	1	1	1	0	0
6	1	1	1	0	1	1	1
5	1	1	1	0	1	1	0
4	1	1	0	1	1	0	0
3	1	0	1	1	1	1	0
2	1	0	1	1	0	1	1
1	0	0	0	1	1	0	0

- Да се научи перцептронът да разпознава цифрата 0 означава само първия ред от таблицата да дава 1 на изхода. Да се научи перцептронът да разпознава цифрата 1 означава само последният ред да дава 1 на изхода на перцептрона.

- С която и цифра да започнете процедурата за конвергенция на перцептроните, цикълът през примерите прави подходящи настройки на тегловния вектор докато той прави грешки и докато по-нататъшни промени не са необходими.

- Когато перцептронът се учи да идентифицира 0, конвергенцията е бърза – необходими са само две промени. Първо защото перцептронът веднага **сгрешава като идентифицира 0 (дава 0 вместо 1**). Нулевият вектор е **добавен** към всички нулеви идентифициращи тегла и е (01111111). След това на следващия пример перцептрона отново **сгрешава при идентифицирането на 9. (дава 1 вместо 0**). Така при втория пример от първото минаване изваждаме тегловния вектор (01111111) от входния вектор за цифрата 9 (11111101) и получаваме **(-10000010), който е задоволителен**. Той идентифицира 0, а всички други цифри се отхвърлят.

- Интересно е, че този резултат означава, че е необходимо да се следят само два сегмента – означените с 0 и 6 за да се открие дали имаме 0. Това се вижда лесно и от фиг. 4

- Когато перцептронът се учи да идентифицира 8 обаче конвергенцията е по-бавна отколкото тази за другите цифри. Изискват се 65 промени за да се създаде крайния тегловен вектор (3 3 0 6 –1 –7 4 -7).
- Реално никога не се знае колко бързо ще се достигне до решението защото дори ако се знае, че съществува задоволителен тегловен вектор не се знае какъв е този вектор. **Не се знае предварително размерът на δ за този вектор.**

ИМА ПРОСТИ ЗАДАЧИ, КОИТО ПЕРЦЕПТРОНИТЕ НЕ МОГАТ ДА ИЗПЪЛНЯТ

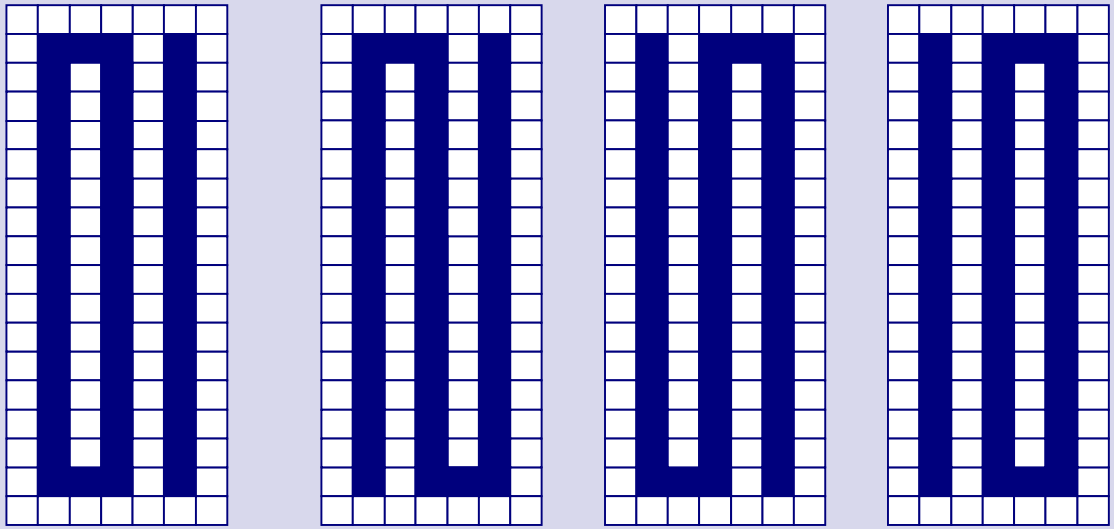
Въпреки, че процедурата за конвергенция на перцептрон гарантира успех когато има решение е важно да разберем, че може да няма решение дори за задачи, които изглеждат прости на пръв поглед.

Между простите задачи, които перцептроът не може да изпълни най-добре позната е задачата за разпознаване на свързаността.

За да разпознае свързаността перцептронът би трябвало да показва, че вторият и третият пример на фиг. 7 са примери на свързани модели, както и, че първият и четвъртият не са.

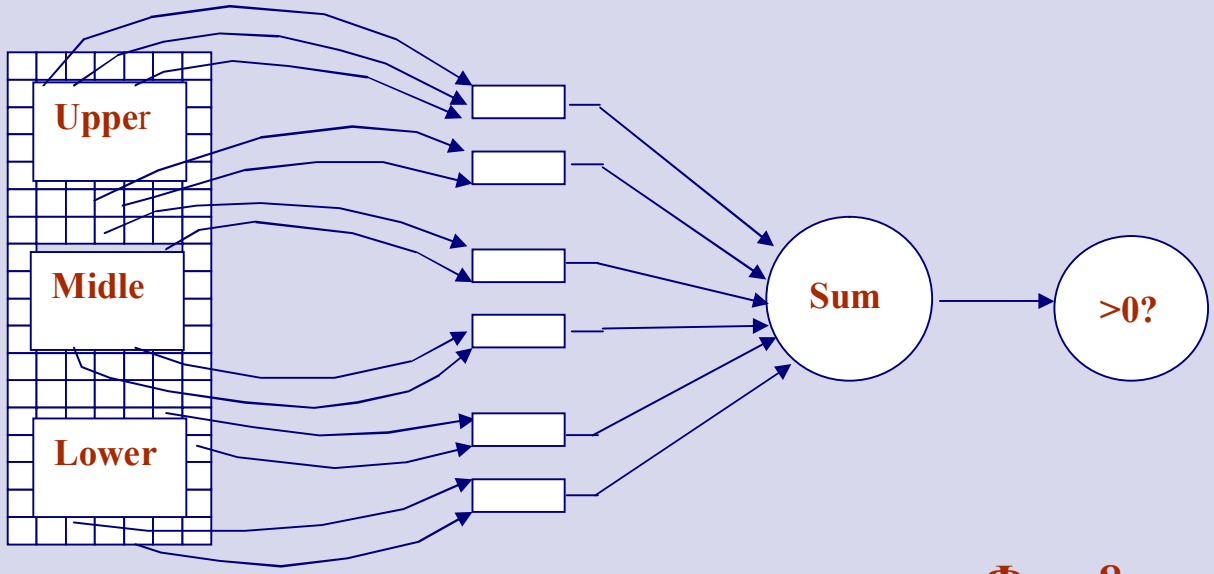
Вторият и третият пример са свързани защото можем да обходим последователно всички черни квадрати като минаваме само по сини квадрати.

Докато при първия и четвъртия пример можем да обходим сините квадрати само като преминаваме през белите.



Фиг. 7

- **Разглеждаме диаметърно – ограничения перцептрон показан на фиг. 8.** Диаметърно – ограниченият перцептрон наблюдава ретината (решетката) като няколко логически блока наблюдават горната част, други следят долната и трети средната като не виждат нищо от другите две съответни части.
- **Този перцептрон не може да реши поставената задача.**
- **Няма индивидуален логически блок, който може да следи промените и в горната и в долната част на мрежата**, което е директно следствие от факта, че перцептронът е диаметърно ограничен.



Фиг. 8