»Лекционен курс

»Изкуствен интелект

Невронни мрежи

Биологични неврони

Биологичните нервни системи обработват информация с помощта на нервни клетки, наречени неврони, които предават съобщения на други клетки, използвайки връзки, наричани синапси.

Информацията в мозъка се съхранява чрез укрепване на синапсите между група неврони чрез процес на подсилване, който многократно обменя съобщения между тези клетки.

Биологични неврони

Невроните работят в групи и реагират на приложените към тях входни данни чрез по-нататъшно разпространение на тези входни данни към други неврони само през подсилените синапси.

Нито един неврон не съхранява определена част от информацията, но група неврони заедно с установени синапси представляват информация в мозъка.

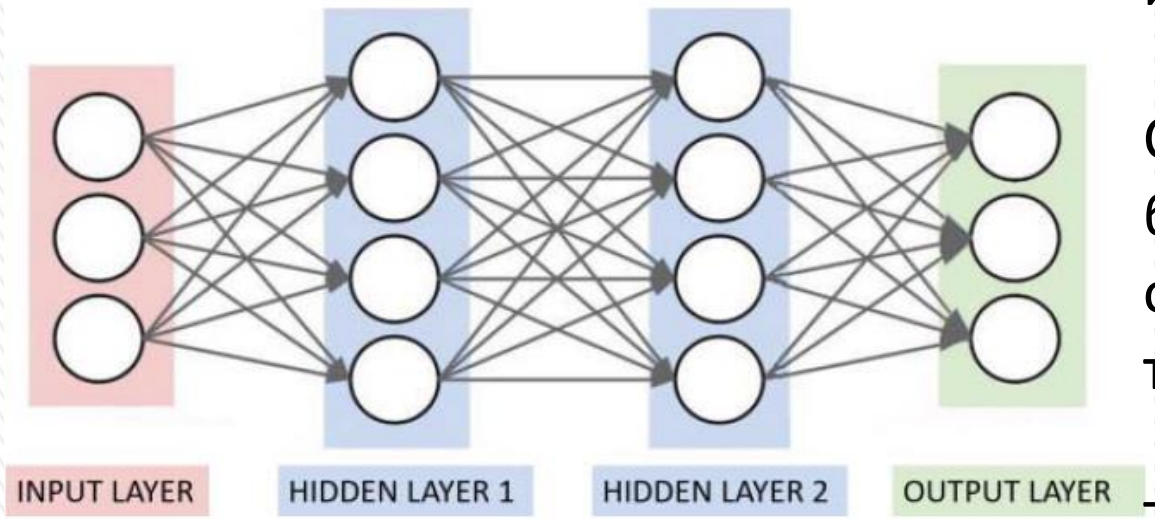
Изкуствени невронни мрежи (ANN)

- Системите, вдъхновени от мозъка, наречени изкуствени невронни мрежи (ANN), възнамеряват да копират начина, по който хората учат.
- ANN са изградени от възли (т.е. изкуствени неврони), които са свързани помежду си чрез насочени връзки (т.е. синапси), представляващи връзка от изхода на един изкуствен неврон към входа на друг.
- Всяка връзка в тази система е свързана с тегло, което променя информацията, докато тя пътува от един неврон към друг.

Слоеста архитектура

- В ANN възлите са организирани в слоеве, отговорни за последователната обработка на информацията.
- Сигналите се обработват, като се започне от входния слой, преминавайки междинните скрити слоеве до достигане на изходния слой, който генерира видими изходи.

Пример



Фигурата показва пример на ANN, състояща се от входен, два скрити и един изходен слой.

Специалистът, проектиращ ANN, избира броя на възлите и слоевете в зависимост от изискванията на проблема, който трябва да бъде разрешен.

Този процес на проектиране изисква опит и специфични познания в областта за всяко приложение.

Приложение на ANN

- ANN могат да бъдат използвани при решаването на много инженерни проблеми, вариращи от киберсигурността до обработката на изображения.
- Като цяло, приложенията на ANN попадат обикновено в една от трите категории:
  - Класификация.
  - Прогнозиране.
  - Оптимизация.

Примери класификация

- При проблемите с класификацията наблюдението от реалния живот се разпознава като определена категория.
- Един пример е създаването на ANN за класифициране на дейности на кредитна карта като легитимни или измамни.
- В този пример ANN може да бъде изградена чрез легитимни действия върху дадена кредитна карта въз основа на историческите данни за употреба на тази карта (напр. видове закупени артикули, посетени места, часове на пазаруване и т.н.).
- Когато едно взаимодействие не се вписва в класа на легитимните действия, то ще бъде маркирано като подозрителна дейност.
- Друг пример за проблем с класификацията включва разпознаване на ръкописни знаци като специфични букви или цифри

Примери прогнозиране

- Проблемите с прогнозирането имат за цел да оценят неизвестни резултати въз основа на исторически данни.
- Например, ANN може да бъде изградена с помощта на минали атмосферни данни за прогнозиране на бъдещи модели на валежи и температурни нива.
- По подобен начин при прогнозите на фондовия пазар, като се използват исторически данни (напр. скорошно пазарно поведение, геополитически ситуации и цени на суровините), може да се настрои ANN за да се предвиди дали дадена акция се очаква да спечели или да загуби стойност.

Пример оптимизация

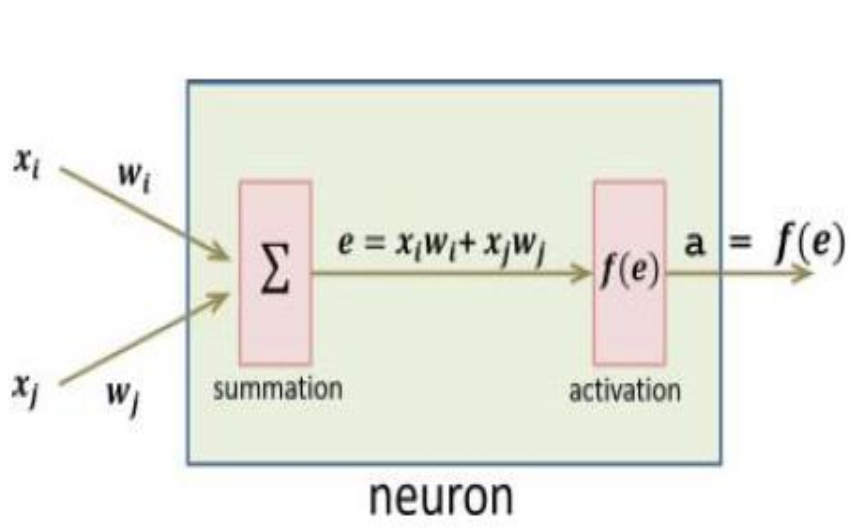
- При проблемите с оптимизацията валидни резултати, които често трябва да удовлетворяват множество и евентуално противоречиви ограничения, трябва да бъдат намерени обикновено в много големи пространства за решения.
- Например, намиране на оптимална доза от химиотерапевтично лекарство, което да се приложи на пациент с рак въз основа на лични данни за туморен растеж и цялостни здравни характеристики (напр. сърдечно състояние, измервания на кръвната захар, физиологични показатели и други) могат да бъдат адресирани чрез ANN от оптимизационен тип.

Компоненти на една ANN

Функции в един неврон

- В ANN всеки възел (т.е. неврон) извършва изчисление, за да определи дали неговите входни сигнали трябва да бъдат препратени към следващия слой.
- Тази задача се изпълнява от така наречените функции за сумиране и активиране.
- Първо, функцията за сумиране събира силата на входящия сигнал и след това функцията за активиране взема решение дали да ги разпространи по-нататък към други слоеве на ANN.

Математически модел

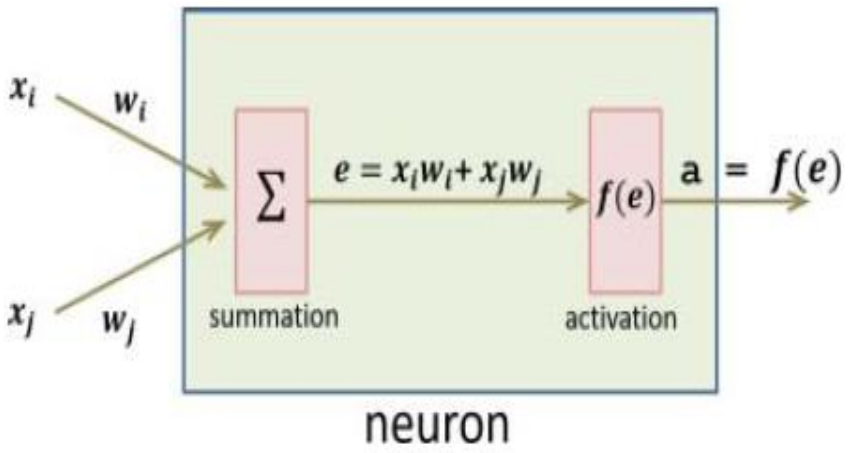


Пример за функции за сумиране и активиране, вградени в изкуствен неврон, са показани на фигурата, където входовете на x_i и x_j с тегла w_i и w_j, съответно, първо се сумират като комбиниран сигнал e.

След това функция за активиране f(e) определя дали резултантният сигнал e трябва да бъде препратен към следващия слой.

Най-често срещаните функции на активиране са под формата на сигмоидни или праволинейни функции.

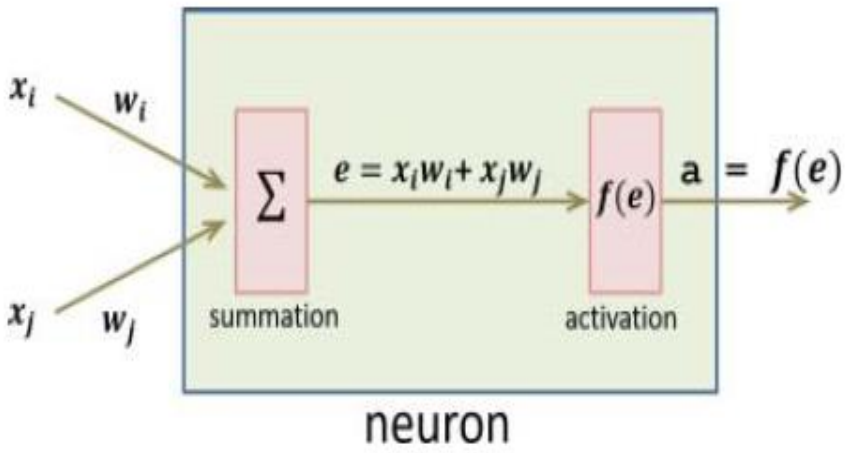
Математически модел



Най-често срещаните функции на активиране са под формата на сигмоидни или праволинейни функции.

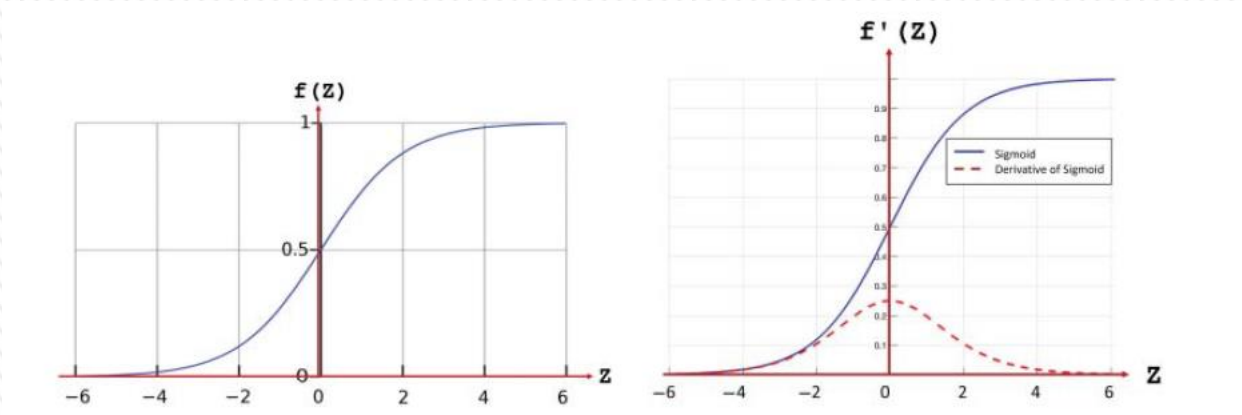
Функцията за сигмоидно активиране трансформира своя вход в изход, вариращ между 0.0 и 1.0, което е полезно, когато изходът е вероятностен.

Математически модел



Праволинейната функция за активиране, от друга страна, генерира изходен сигнал, равен на силата на входния сигнал за всички положителни стойности на сигнала, и извежда нула за всеки вход с отрицателни стойности.

Сигмоидна активираща функция



(1) $$f(Z) = \frac{1}{1 + e^{-Z}}.$$ $$f'(Z) = \frac{e^{-Z}}{(1 + e^{-Z})^2}.$$

Сигмоидът е удобен за ефективно изчисляване на градиенти, използвани в обучението на ANN.

Сигмоидното активиране ограничава невронния изход в интервала от [0, 1].

Производни (напомняне)

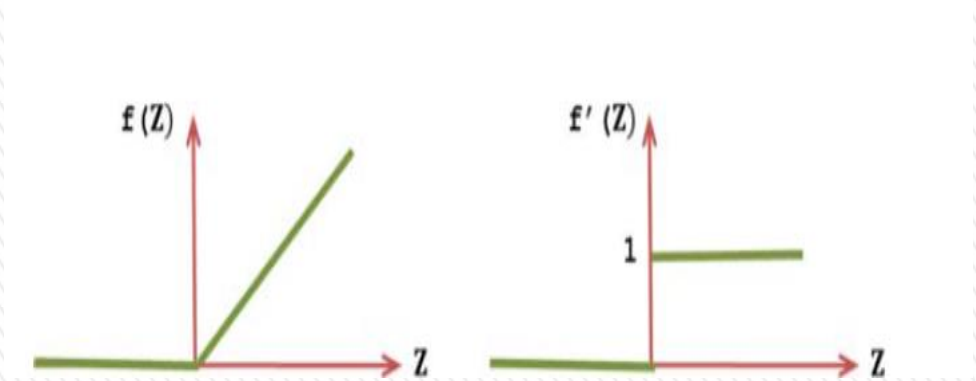
$$y = f(x).g(x) \Rightarrow y' = f'(x)g(x) + f(x)g'(x)$$

If $f(x) = C$ (C е константа), и $y = f(x)g(x)$

$$y = Cg(x)\ y' = C'.g(x) + C.g'(x) = 0 + C.g'(x) = C.g'(x)$$

$$y = \frac{f(x)}{g(x)} \qquad y' = \frac{f'(x)g(x) - f(x)g'(x)}{g^2(x)}$$

Праволинейна активираща функция

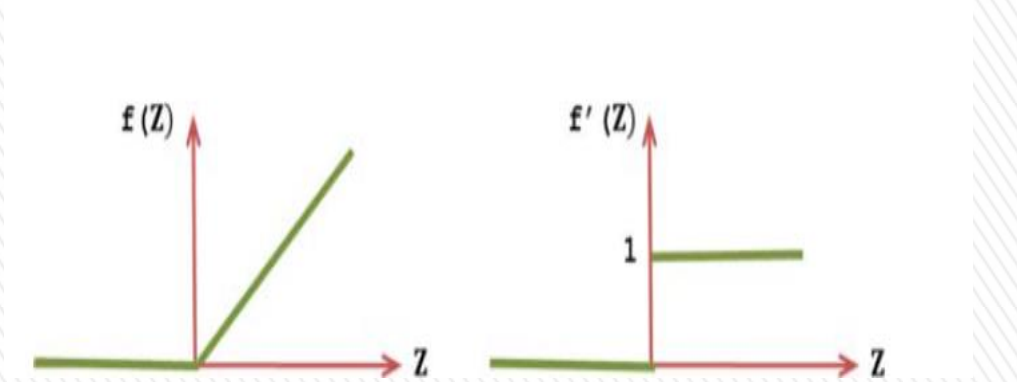


(2) $$f(Z) = \begin{cases} Z & \text{if } Z > 0; \\ 0 & \text{otherwise.} \end{cases} \qquad f'(Z) = \begin{cases} 1 & \text{if } Z > 0; \\ 0 & \text{if } Z < 0; \\ \text{undefined} & \text{if } Z = 0. \end{cases}$$

Праволинейната функция на активиране (ReLU) и нейната производна са показани на фигурата.

По-евтино е да се изчислят градиенти, използвани в обучението на ANN с праволинейно активиране.

Праволинейна активираща функция



$$f(Z) = \begin{cases} Z & \text{if } Z > 0; \\ 0 & \text{otherwise.} \end{cases}$$

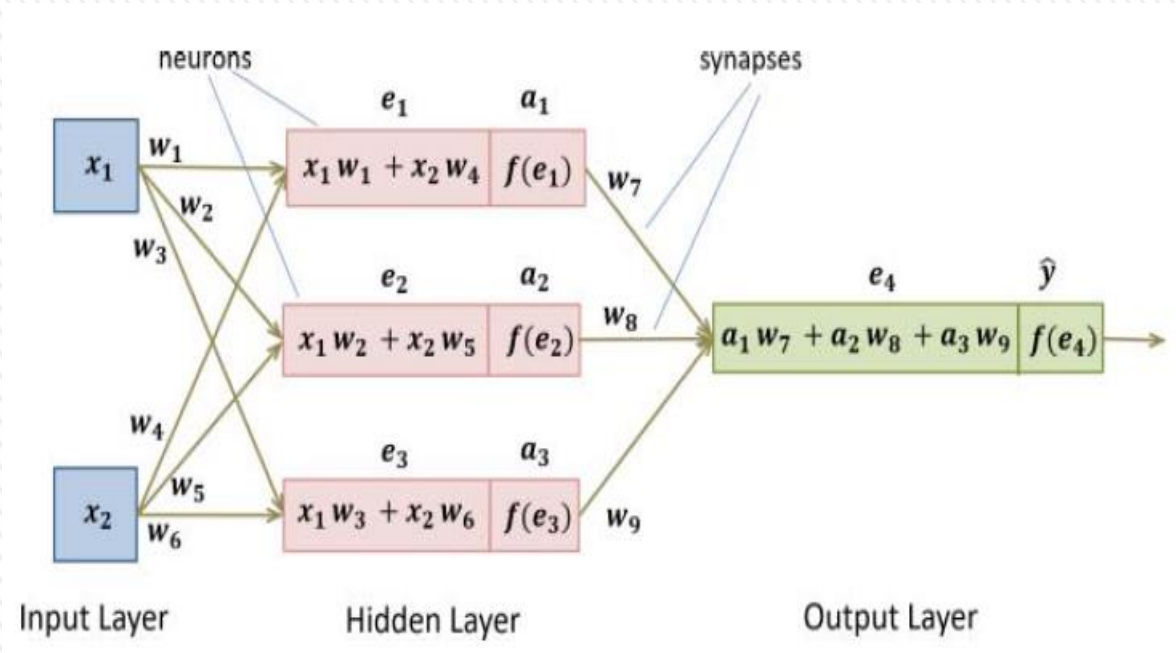
$$f'(Z) = \begin{cases} 1 & \text{if } Z > 0; \\ 0 & \text{if } Z < 0; \\ \text{undefined} & \text{if } Z = 0. \end{cases}$$

Праволинейната функция предотвратява по-нататъшното разпространение на отрицателни сигнали в ANN.

Въпреки това, в изходния слой на ANN, елиминирането на отрицателни стойности чрез праволинейна функция може да бъде проблематично, ако изходните данни на ANN могат да бъдат отрицателни.

В тези случаи в изходния слой не трябва да се използват праволинейни функции за активиране.

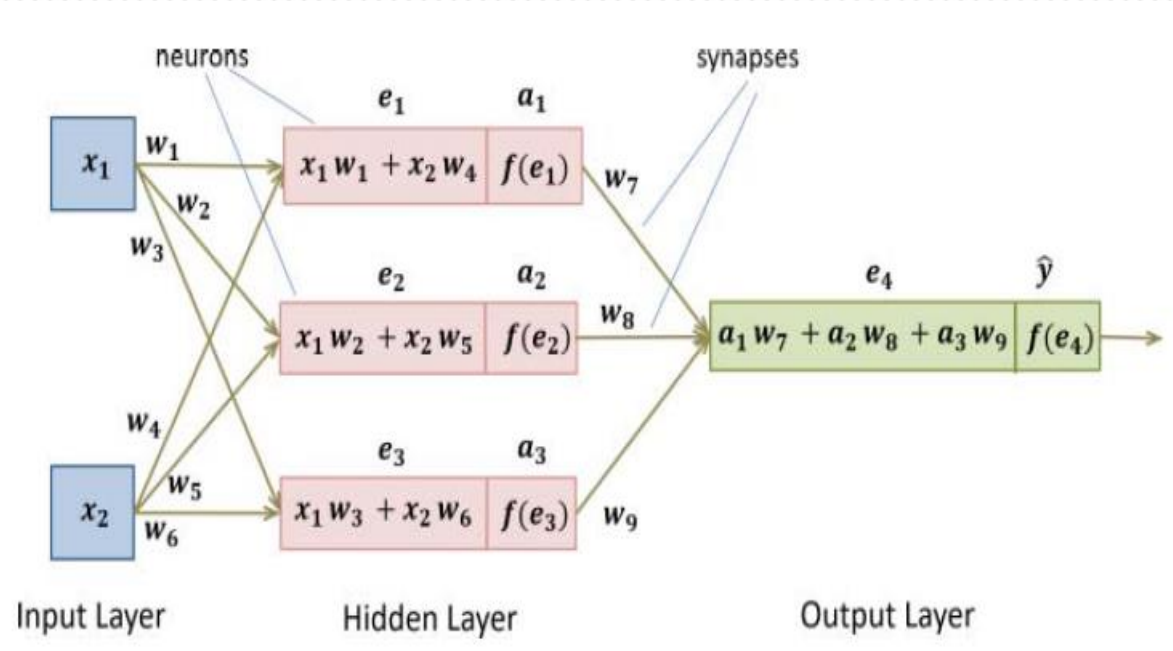
Как работи един неврон?



Стойността на сигнала, след като той напусне междинен неврон, се променя допълнително чрез умножаването му по теглото на връзката, през която се движи.

Фигурата показва примерна ANN, където x_1 и x_2 са входните сигнали към три възела на скрит слой.

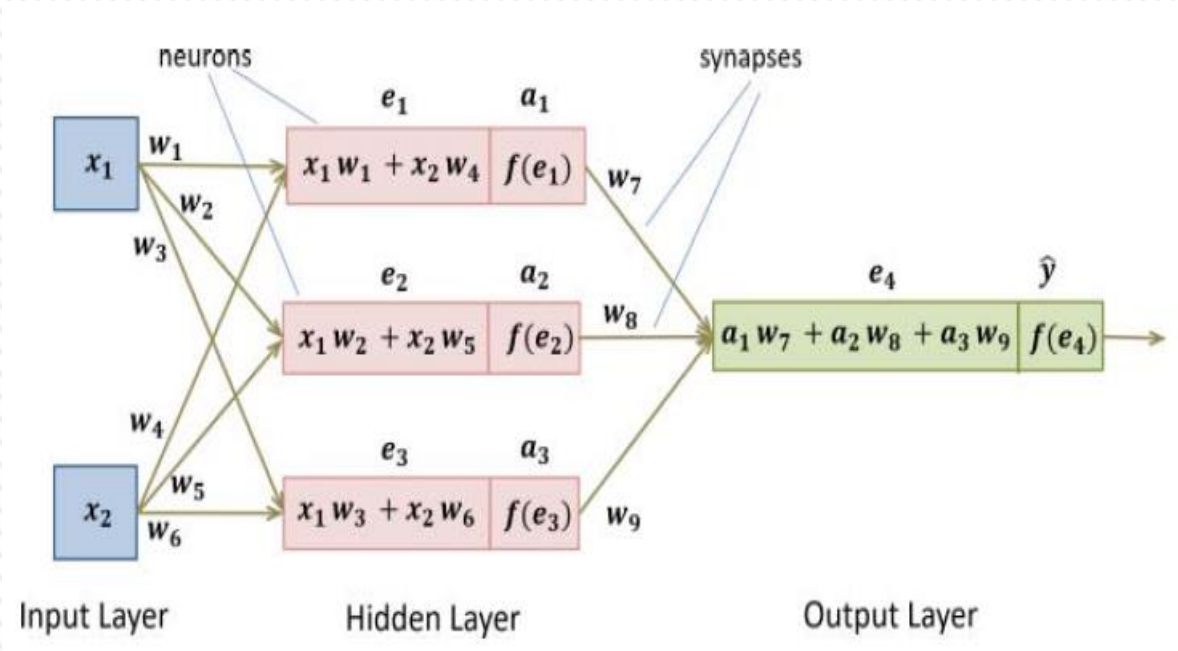
Как работи един неврон?



При трите скрити неврона теглата на входните синапси се умножават по стойностите на входния неврон и се агрегират при скритите неврони, както следва:

$$e_1 = (x_1 \cdot w_1) + (x_2 \cdot w_4)$$
$$e_2 = (x_1 \cdot w_2) + (x_2 \cdot w_5)$$
$$e_3 = (x_1 \cdot w_3) + (x_2 \cdot w_6).$$

Как работи един неврон?

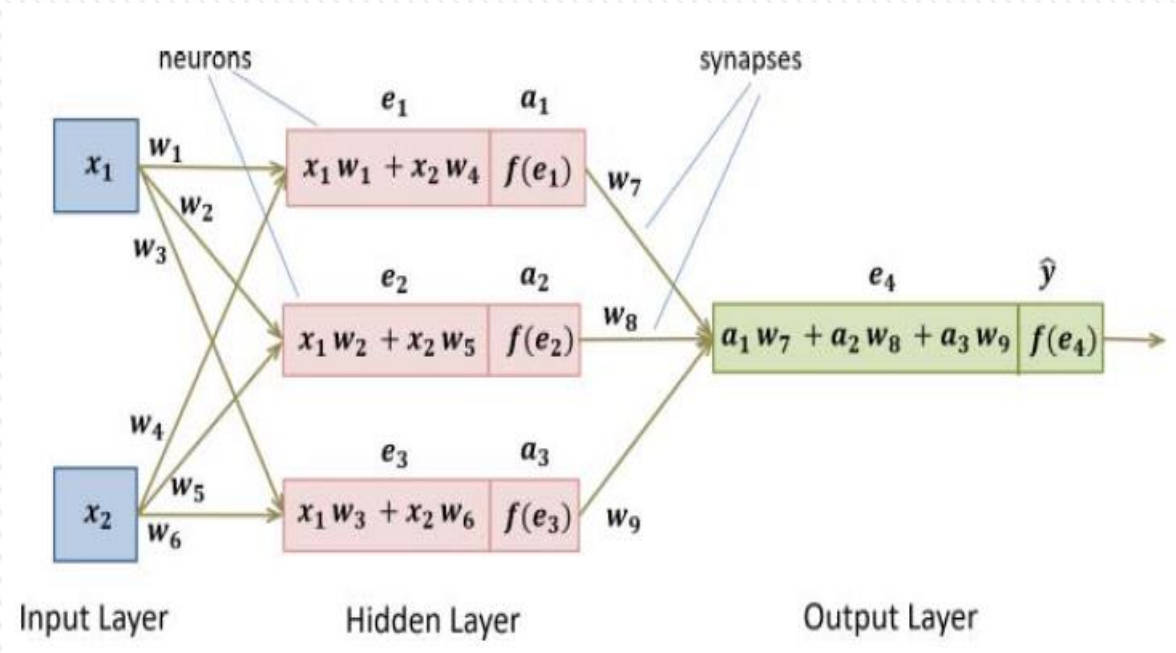


Функция за активиране f(.) се прилага към сумираните сигнали във всеки скрит неврон, както следва:

$$a_1 = f(e_1) \qquad a_2 = f(e_2) \qquad a_3 = f(e_3),$$

където стойностите a_1, a_2 и a_3 се определят въз основа на функцията на активиране, използвана при невроните, както е представено в уравнения (1) и (2).

Как работи един неврон?

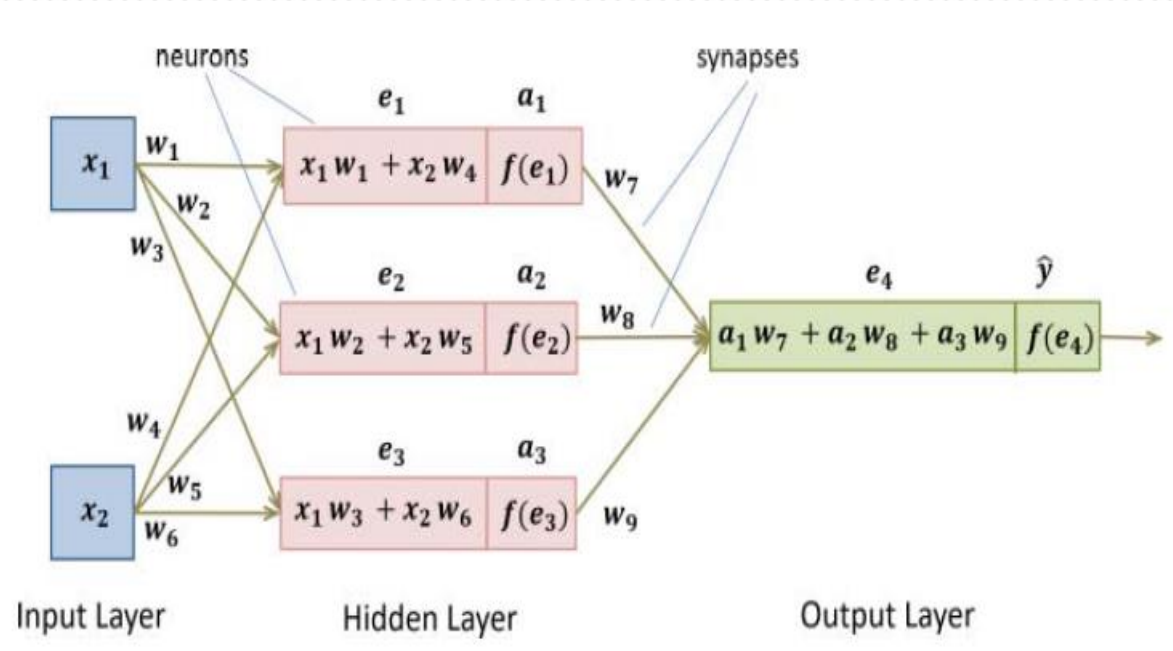


По подобен начин в изходния слой изходният неврон има три входа, които се агрегират като:

$$e_4 = (a_1 \cdot w_7) + (a_2 \cdot w_8) + (a_3 \cdot w_9),$$

където теглата w_7, w_8 и w_9 се прилагат съответно към изходите на скритите неврони a_1, a_2 и a_3.

Как работи един неврон?



Резултатът се получава след прилагане на функция за активиране към сумата, показана в уравнение:

$$\hat{y} = f(e_4).$$

Използване на ReLU

» Както беше обсъдено по-рано, трябва да се внимава, когато се използва праволинейна активираща функция при изходящ неврон, както е показано в уравнението, което позволява само положителни изходни стойности, тъй като отрицателните стойностите може да са важни за определени приложения.

Повече скрити слоеве

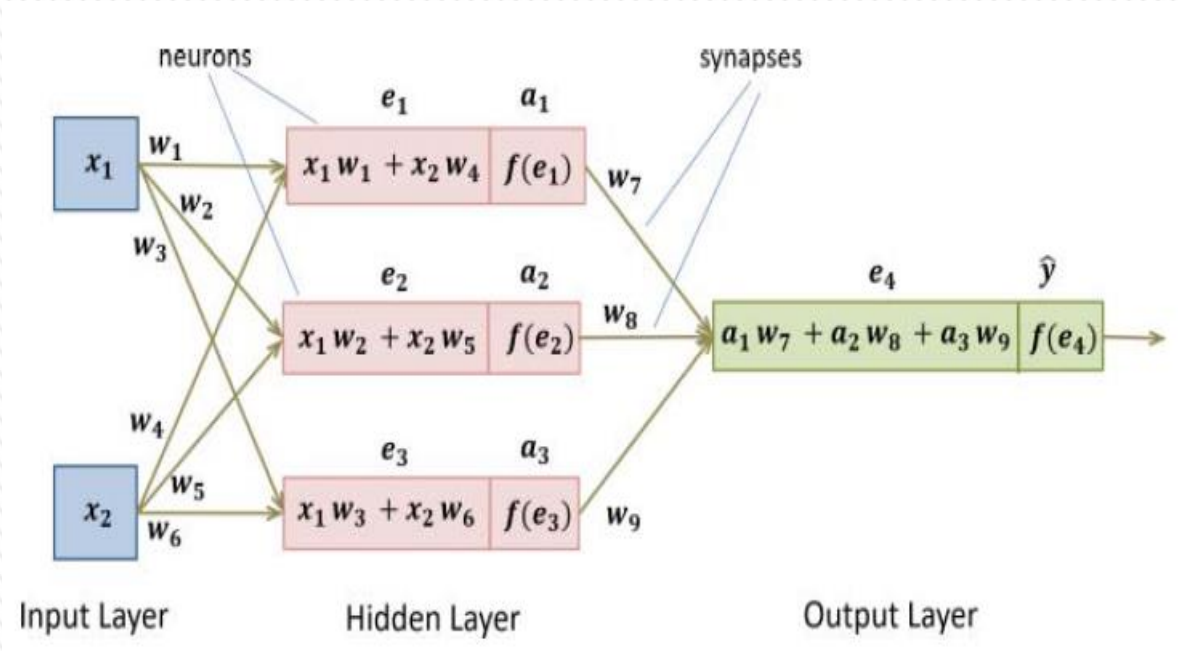
- В общия случай, ако има множество скрити слоеве в ANN, входовете и изходите на последователните скрити слоеве се изчисляват по същия начин.
- Изходният слой може да съдържа повече от един неврон, тъй като ANN може да бъде проектирана за множество изходи в зависимост от приложението.
- Например, ANN за насочване на дрон може да генерира три изхода (скорост, посока и надморска височина) за полетен контролер, използвайки множество входове като текуща позиция, скорост, надморска височина, план на полета и информационни сензори за препятствия.

Обучение на невронни мрежи

Проба-грешка

- За да може една ANN да генерира прогнози, тя първо трябва да бъде обучена с помощта на съществуващи примерни данни по начин, подобен на традиционните подходи проба-грешка.
- По време на обучението теглата, свързани със синапсите (напр. w_1 до w_8 на фигурата), се коригират така, че разликата между изходните данни, генерирани от ANN, и действителните изходни стойности на съществуващите данни да са възможно най-малки.
- Тази разлика се нарича грешка или загуба.
- Целта на тренировката е да се сведе до минимум (или да се намали колкото е възможно повече) загубата чрез коригиране на стойностите на теглото.

Пример

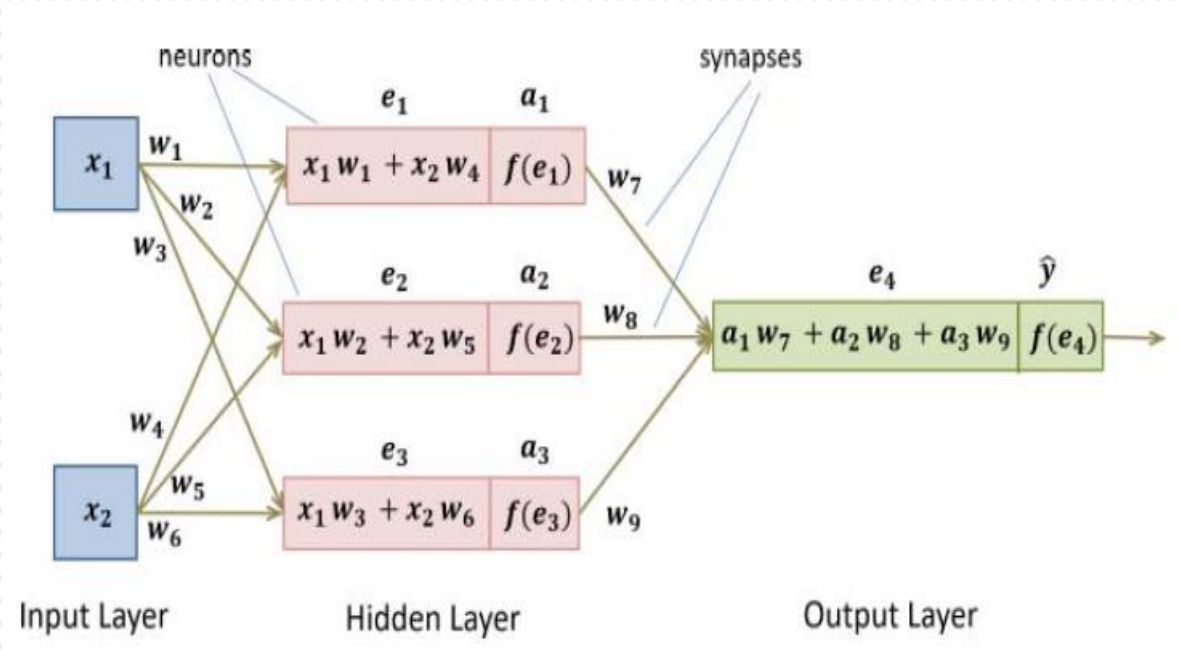


Нека сега илюстрираме процеса на обучение, като използваме примера на ANN, даден на фигурата.

Да предположим, че имаме следните точки от данни, наблюдавани в полето като входове и изходи на събитие, за което ANN трябва да бъде обучена:

input 1: $x_1 = 3$ input 2: $x_2 = 5$ output: $y = 75.$

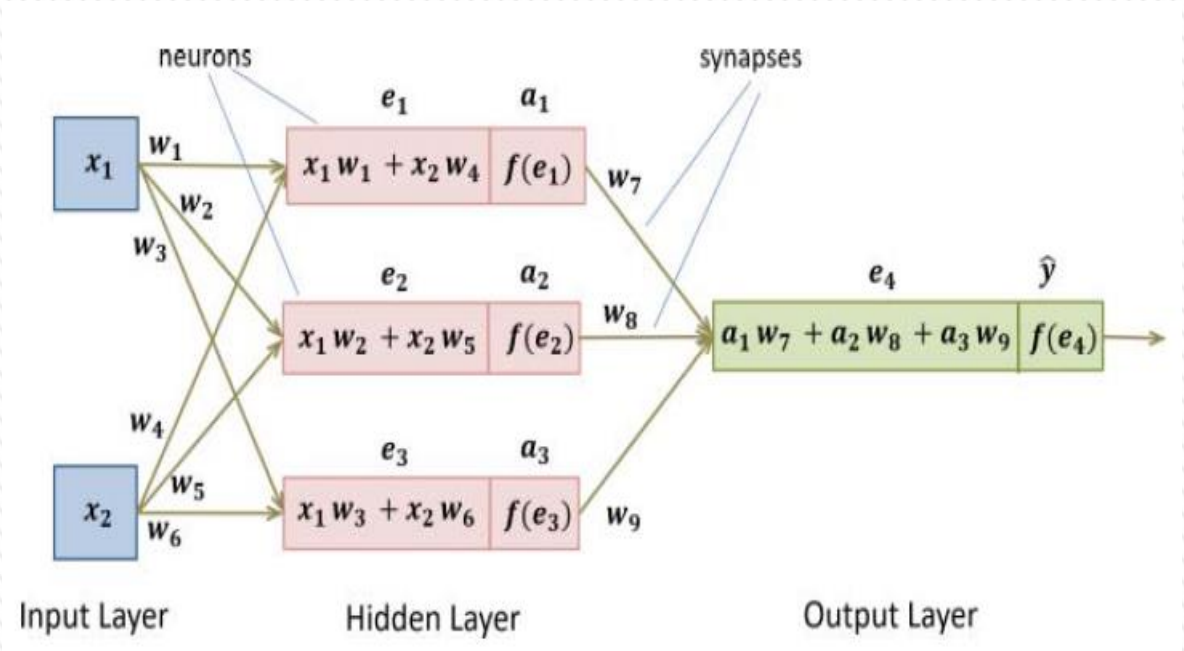
Пример



Да предположим, че текущите начални стойности на теглата на ANN са:

$$w_1 = w_2 = w_3 = 1.0$$
$$w_4 = w_5 = w_6 = 0.5$$
$$w_7 = w_8 = w_9 = 4.0.$$

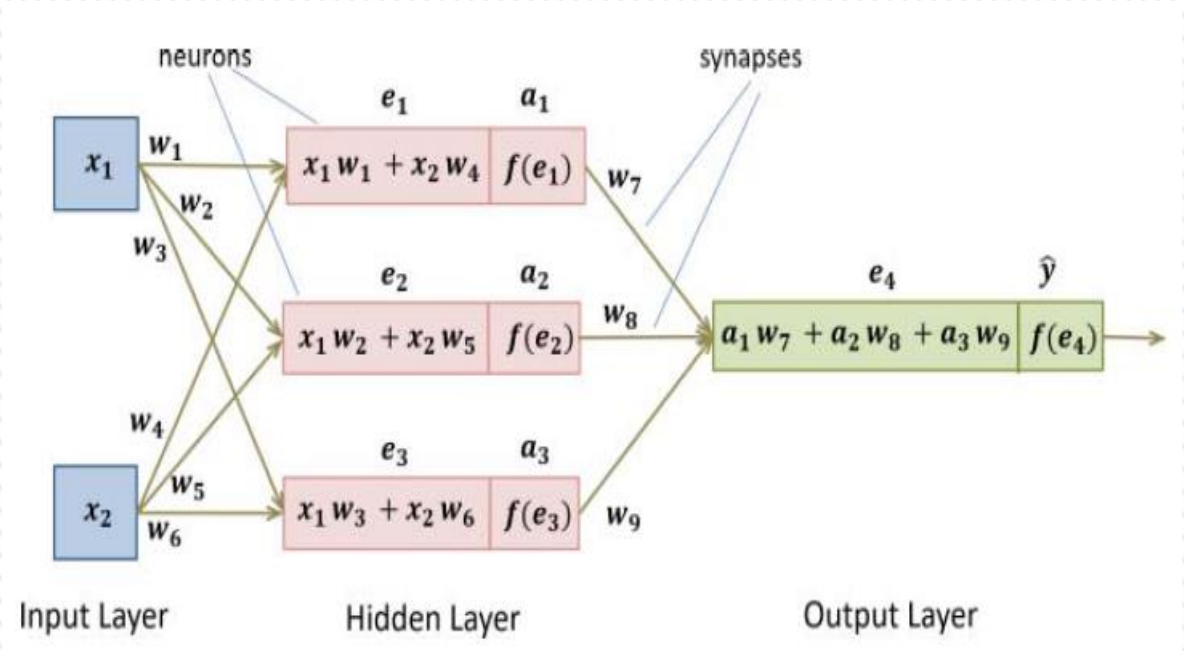
Пример



След това изчисляваме:

$$e_1 = (3 \cdot 1) + (5 \cdot 0.5) = 5.5$$
$$e_2 = (3 \cdot 1) + (5 \cdot 0.5) = 5.5$$
$$e_3 = (3 \cdot 1) + (5 \cdot 0.5) = 5.5.$$

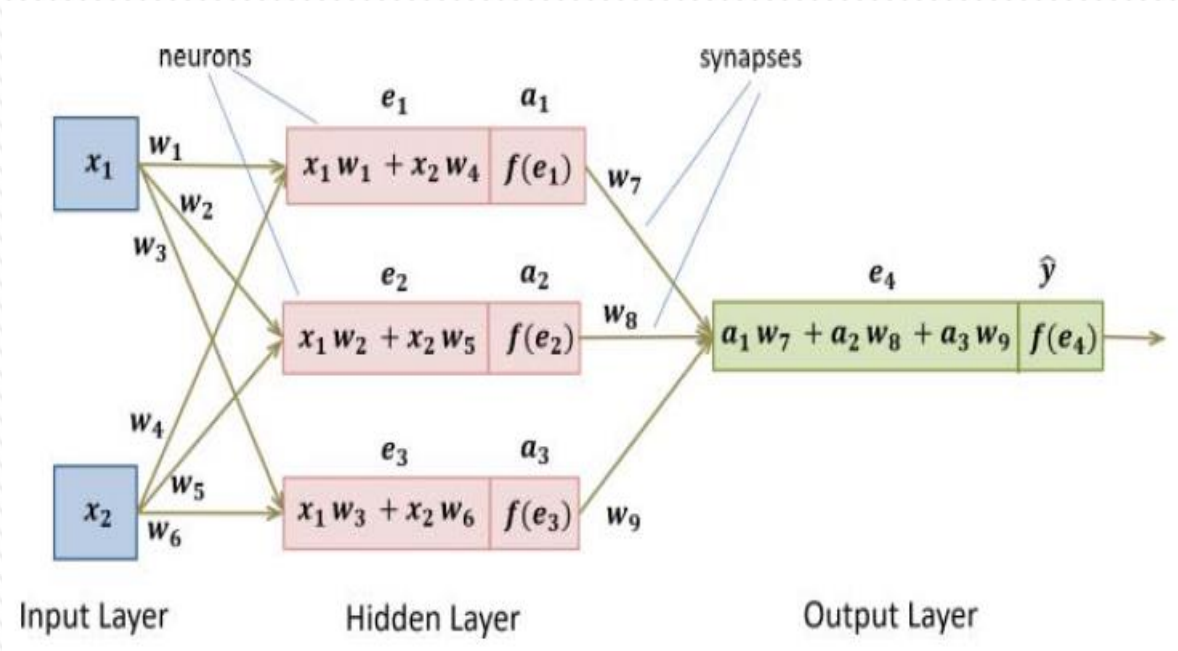
Пример



Използвайки праволинейна активираща функция получаваме:

$$a_1 = 5.5 \qquad a_2 = 5.5 \qquad a_3 = 5.5.$$

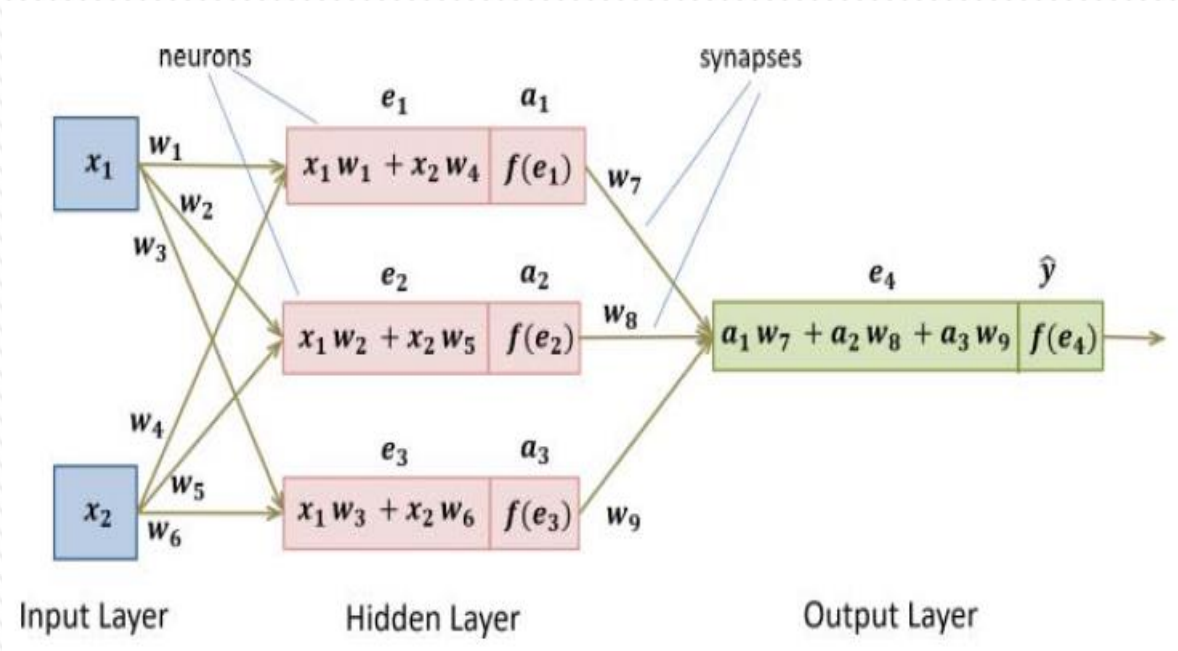
Пример



Използвайки праволинейна активираща функция изчисляваме:

$$e_4 = (5.5 \cdot 4) + (5.5 \cdot 4) + (5.5 \cdot 4) = 66.$$

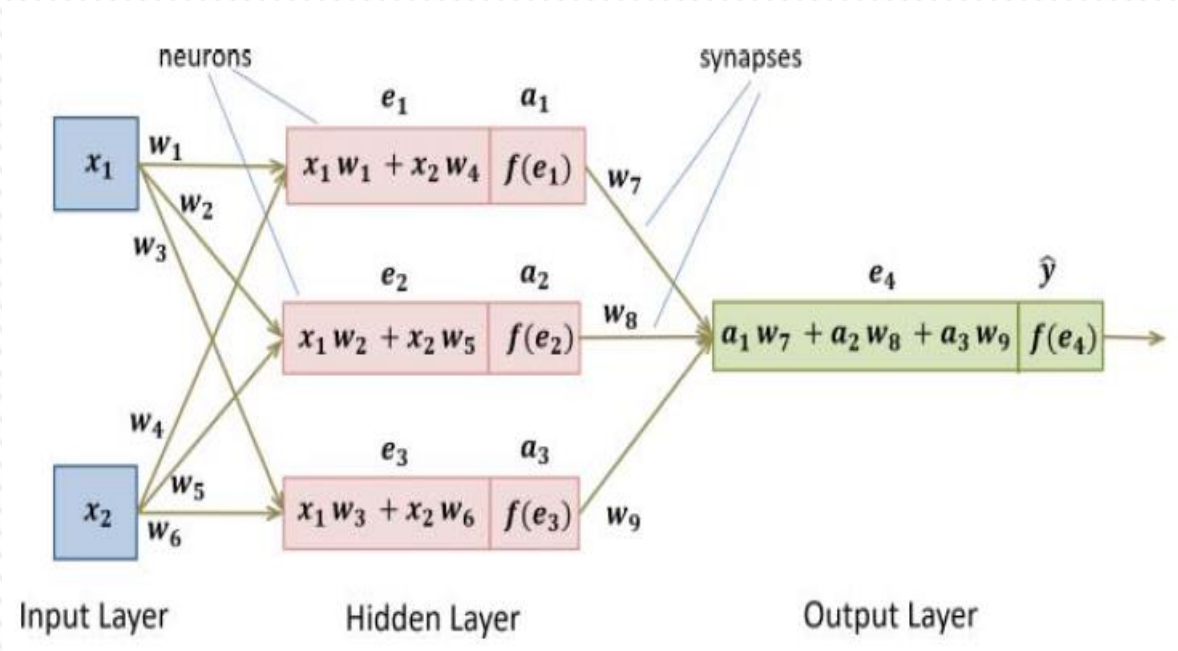
Пример



Т.е., след първата итерция получаваме следната стойност:

$$\hat{y} = 66.$$

Пример

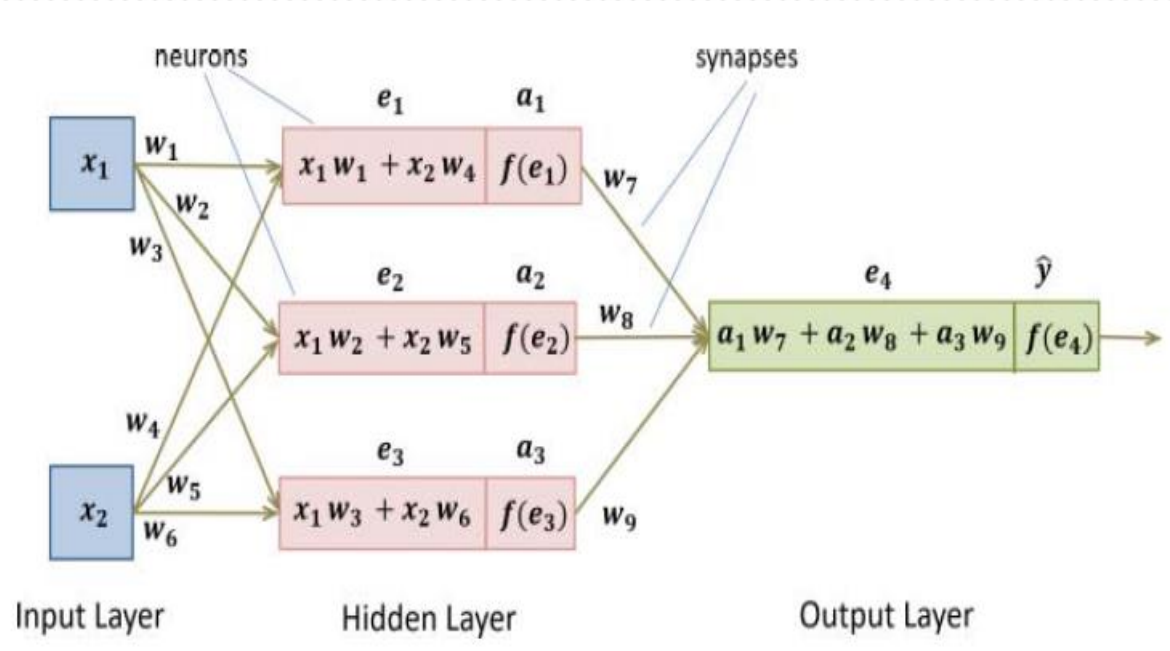


Грешката се определя като разликата между генерирания от ANN изход и реалната изходна стойност от данните за обучение.

За този пример грешката е:

$$\text{error} = y - \hat{y} = 75 - 66 = 9.$$

Пример



За реални приложения са възможни различни методи за изчисляване на грешката, например използване на квадрати на разликите или логаритмични стойности.

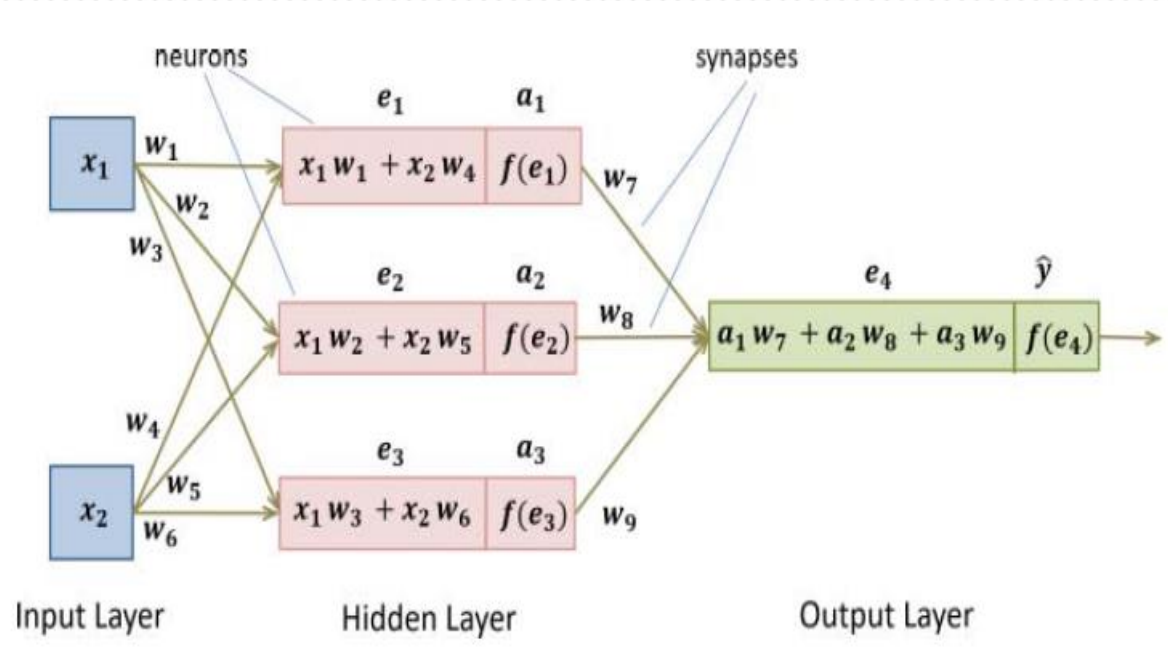
Да предположим, че теглата са коригирани чрез процеса на обучение на ANN и новите стойности на тегла сега са:

$$w_1 = w_2 = w_3 = 1.0$$

$$w_4 = w_5 = w_6 = 0.8$$

$$w_7 = w_8 = w_9 = 3.5.$$

Пример



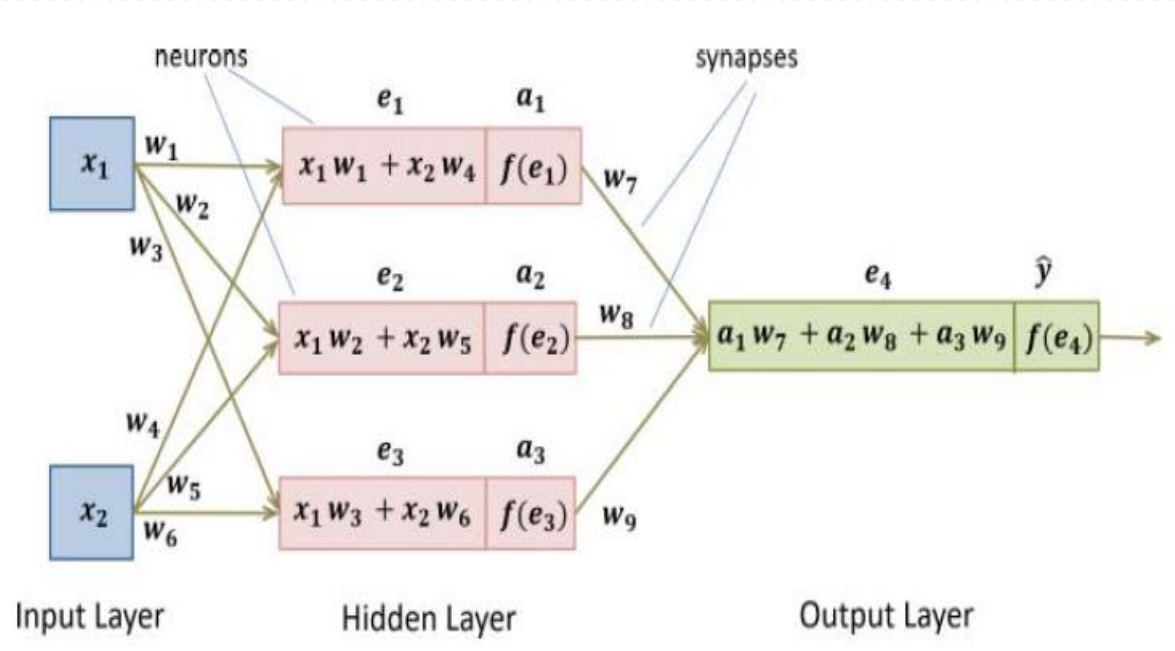
Използвайки тези стойности на теглото с примерните данни, имаме:

$$e_1 = (3 \cdot 1) + (5 \cdot 0.8) = 7$$
$$e_2 = (3 \cdot 1) + (5 \cdot 0.8) = 7$$
$$e_3 = (3 \cdot 1) + (5 \cdot 0.8) = 7,$$

И съответно:

$$e_4 = (7 \cdot 3.5) + (7 \cdot 3.5) + (7 \cdot 3.5) = 73.5.$$

Пример



В този случай разликата между реалната изходна стойност на y и изхода, генериран от ANN на ˆy (т.е. грешката) е:

$$\text{error} = y - \hat{y} = 75 - 73.5 = 1.5.$$

Обобщение

- Можем ясно да видим, че е възможно да се намали грешката чрез коригиране на стойностите на теглата на синапсите, което е основната идея на процеса на обучение на ANN.
- ANN се обучава чрез използване на входните и изходните данни (наричани също исторически, примерни или полеви данни), за да присвои правилни стойности на теглата на синапсите (връзките).

Процес на обучение (обобщение)

- За да започнем, на теглата на връзката се присвояват произволно избрани стойности.
- Използвайки тези произволни стойности, изходните стойности се изчисляват чрез умножаване на входовете и теглата и прилагане на функциите за активиране към всеки неврон на ANN.
- Стъпките до тази точка се наричат разпространение напред.

Процес на обучение (обобщение)

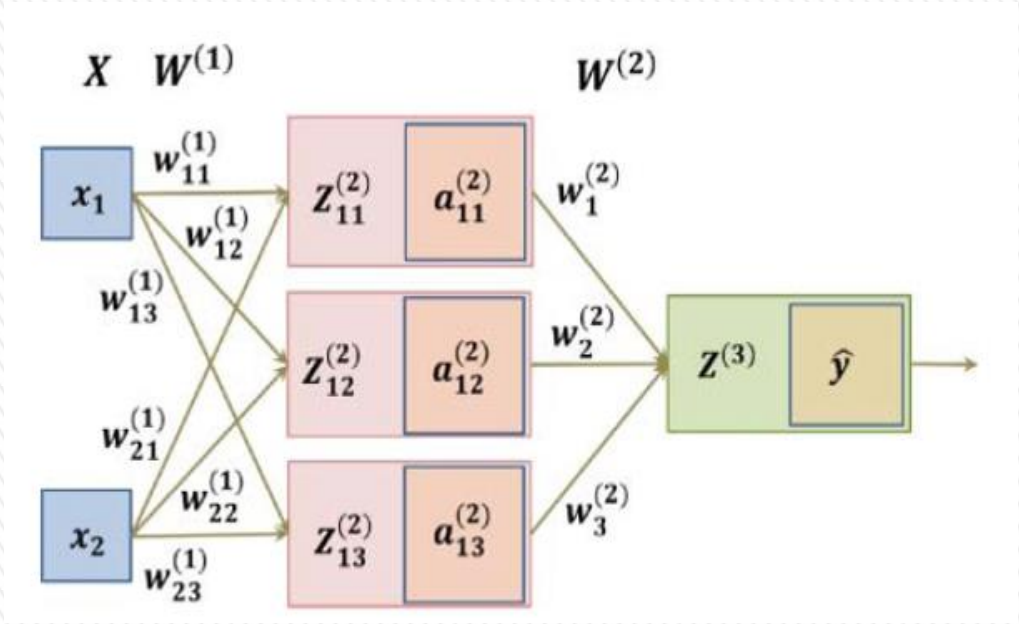
- След това резултатите се сравняват с очакваните стойности от примерните данни - разликата между генерираните от ANN и реалните изходни стойности е грешката, генерирана от ANN.
- След разпространението напред и изчисляването на загубата се извършва обратно разпространение въз основа на частични производни на загуба по отношение на теглата на ANN на всеки слой.
- Въз основа на частичните производни стойностите на теглата се актуализират.

Процес на обучение (обобщение)

- Използвайки новите тегла, последователните стъпки на разпространение напред и назад (наречени епохи) се изпълняват както по-горе.
- Краят на обучението се определя от проектантите въз основа на различни критерии, включително извършване на предварително определен брой епохи, стойността на загубата, достигаща желана ниска стойност или липса на напредък от епоха на епоха по отношение на намаляването на загубите.

Разпространение напред

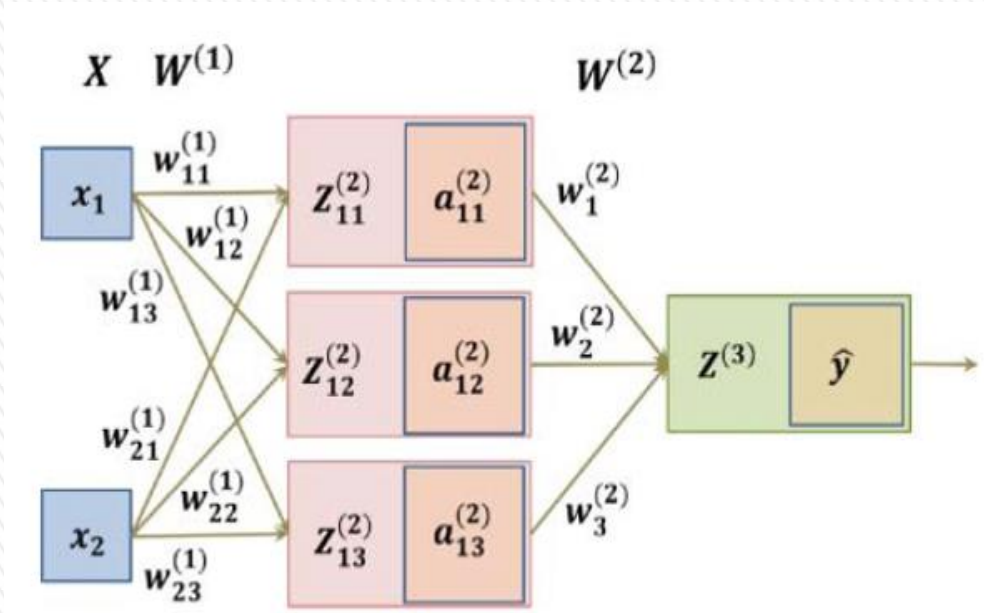
Примерна мрежа



Нека сега проучим стъпката на разпространение напред на процеса на обучение на ANN.

Ще използваме проста ANN с един скрит слой, както е показано на фигурата.

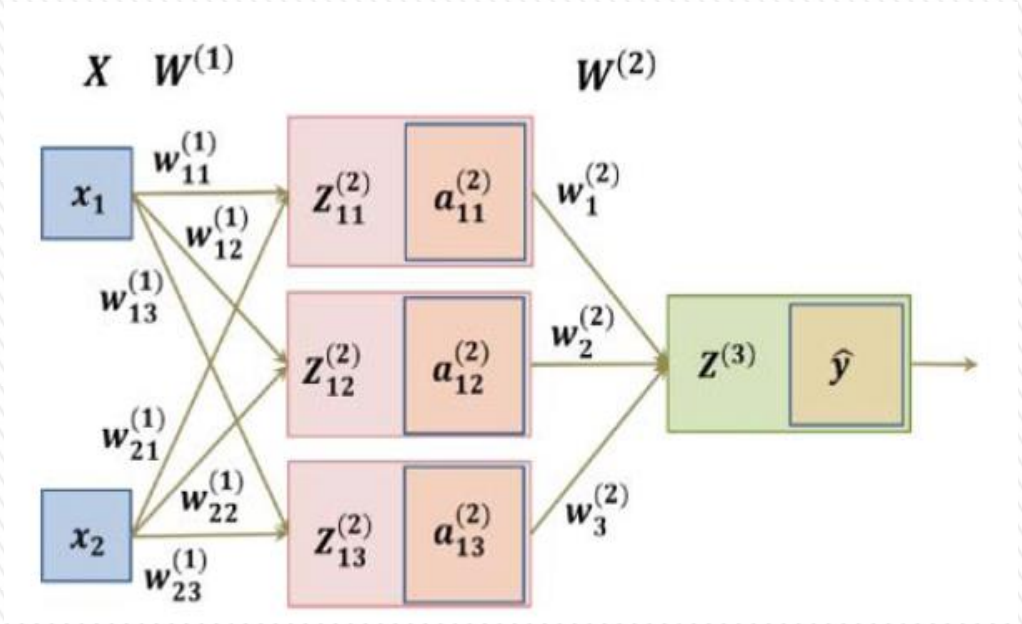
Примерна мрежа



За общия случай, когато ANN имат множество входове, изходи и скрити слоеве, деривациите са валидни със съответно увеличени размери на матрицата на теглото.

Да си спомним, че всеки синапс на ANN има независимо тегло, което се умножава по изхода на началния неврон.

От вход към скрит слой



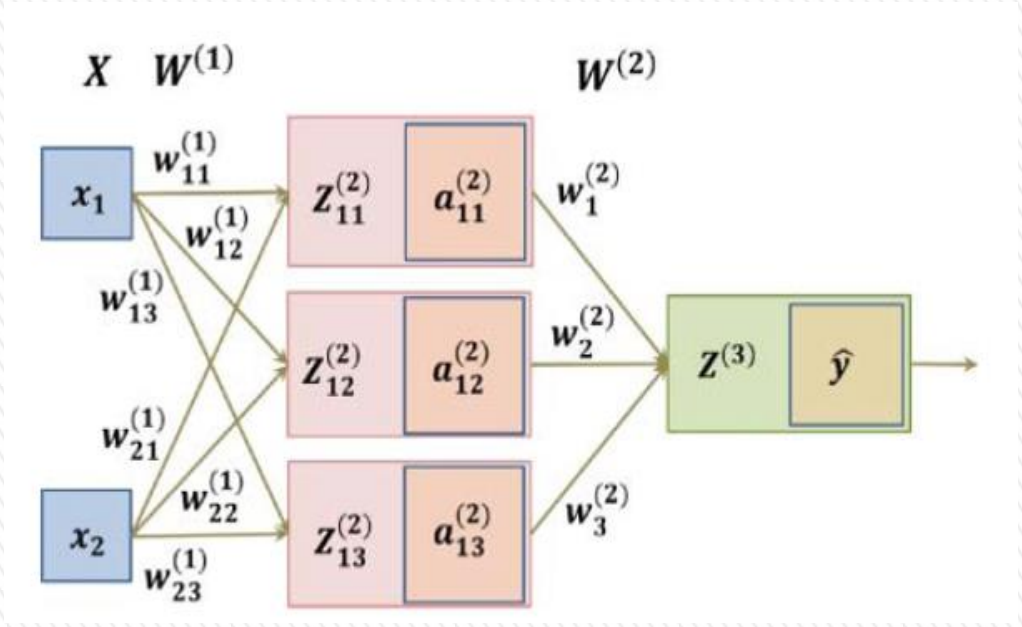
За разпространение напред от вход към скрит слой на ANN, показана на фигурата, имаме:

$$Z^{(2)}_{(11)} = x_1 \cdot w^{(1)}_{11} + x_2 \cdot w^{(1)}_{21}$$

$$Z^{(2)}_{(12)} = x_1 \cdot w^{(1)}_{12} + x_2 \cdot w^{(1)}_{22}$$

$$Z^{(2)}_{(13)} = x_1 \cdot w^{(1)}_{13} + x_2 \cdot w^{(1)}_{23}.$$

От вход към скрит слой



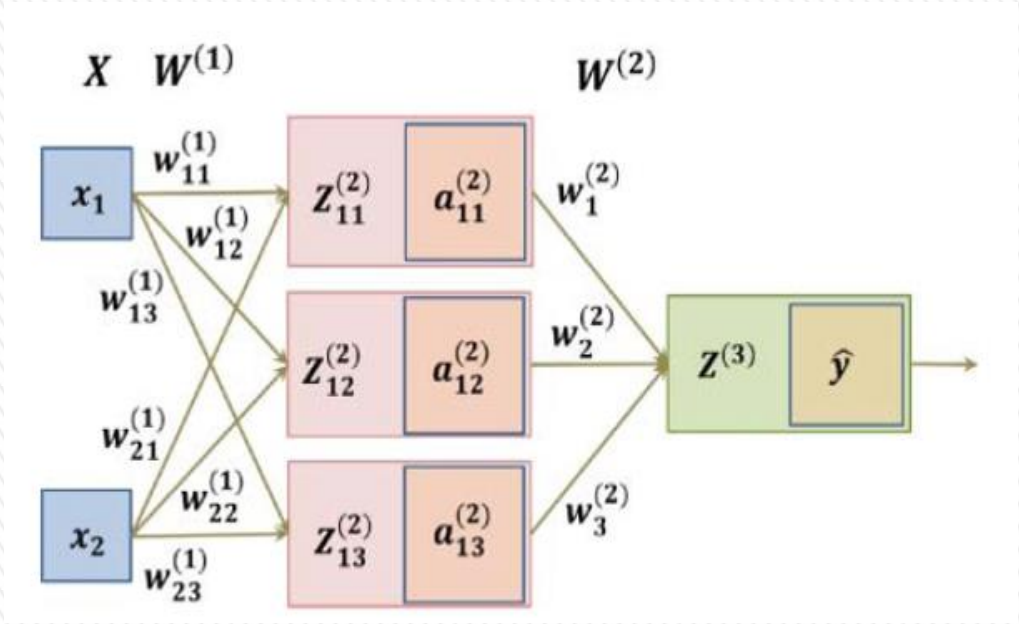
Използвайки определението за сигмоидно активиране имаме следните изходи от невроните на скрития слой:

$$a^2_{(11)} = f(Z^{(2)}_{(11)}) = \frac{1}{(1+e^{-(x_1 \cdot w^{(1)}_{11} + x_2 \cdot w^{(1)}_{21})})^2}$$

$$a^2_{(12)} = f(Z^{(2)}_{(12)}) = \frac{1}{(1+e^{-(x_1 \cdot w^{(1)}_{12} + x_2 \cdot w^{(1)}_{22})})^2}$$

$$a^2_{(13)} = f(Z^{(2)}_{(13)}) = \frac{1}{(1+e^{-(x_1 \cdot w^{(1)}_{13} + x_2 \cdot w^{(1)}_{23})})^2}.$$

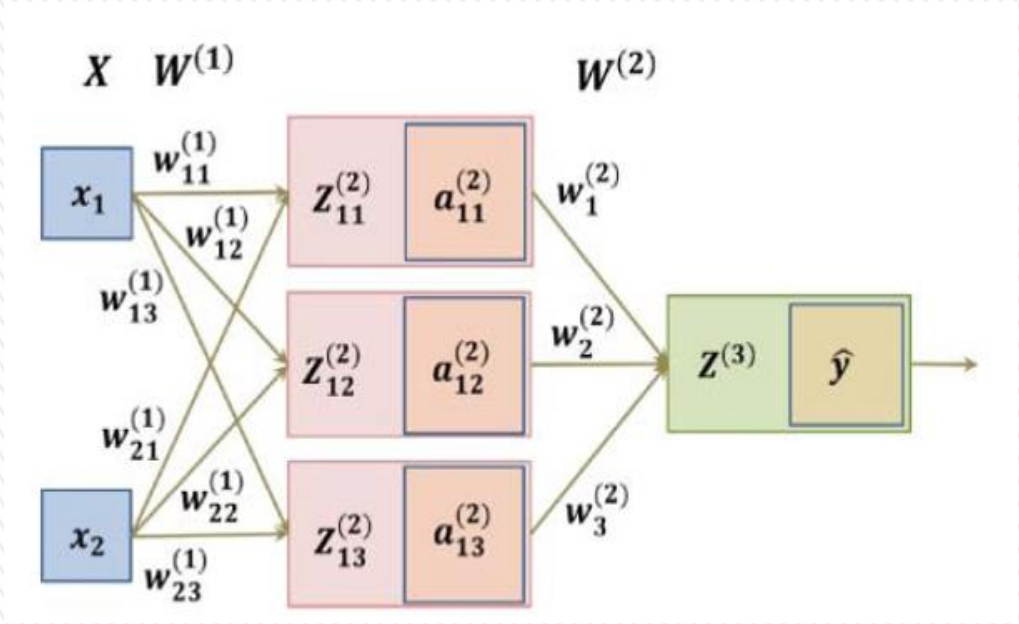
От скрит към изходен слой



При неврона на изходния слой се получава агрегиране на трите входни синапса:

$$Z^{(3)} = a_{11}^{(2)} \cdot w_1^{(2)} + a_{12}^{(2)} \cdot w_2^{(2)} + a_{13}^{(2)} \cdot w_3^{(2)}.$$

От скрит към изходен слой



И накрая, на изходния слой, използвайки сигмоидна функция за активиране, генерира ˆy като:

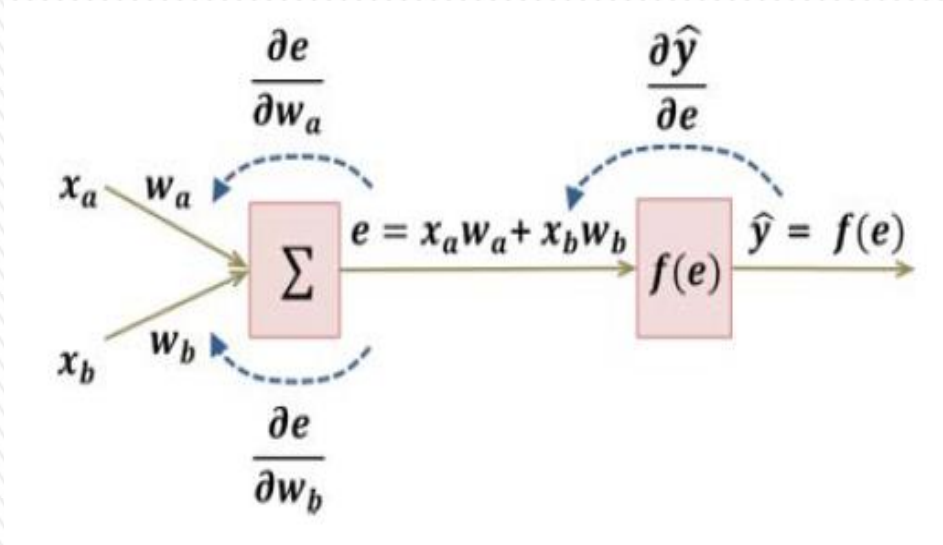
$$\hat{y} = f(Z^{(3)}) = \frac{1}{1+e^{-Z}} = \frac{1}{1+e^{-(a_{11}^{(2)} \cdot w_1^{(2)} + a_{12}^{(2)} \cdot w_2^{(2)} + a_{13}^{(2)} \cdot w_3^{(2)})}}.$$

Обратно разпространение

Необходимост

- След като се извърши разпространение напред, грешката трябва да се разпространи обратно, като се започне от изходния слой чак до входния слой, така че теглата на синапса да могат да се актуализират въз основа на стойностите на грешката.
- Нека проучим как работи процесът на обратно разпространение за даден неврон.

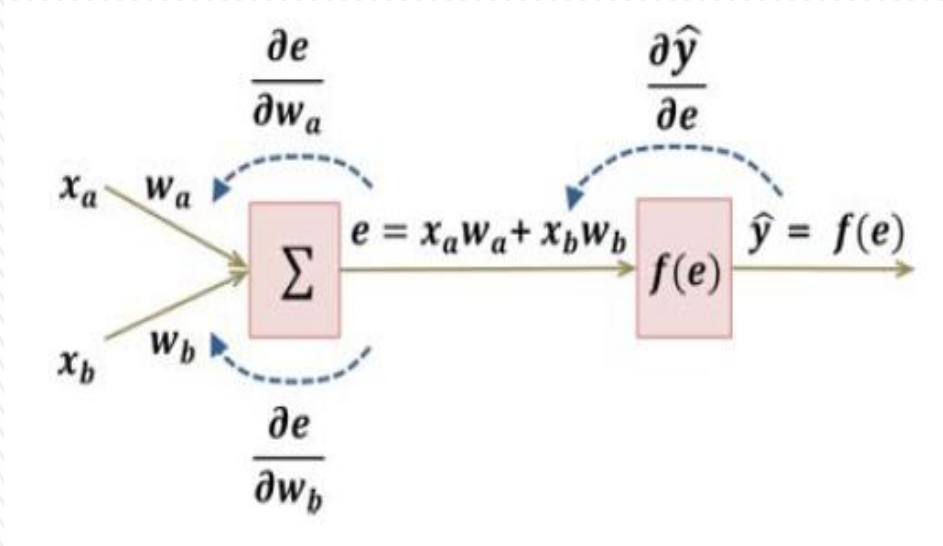
Пример



Да предположим, че има два входа, а именно x_a и x_b, върху две входящи връзки с тегла w_a и w_b, както е показано на фигурата.

Нека изходът на този неврон е ˆy = f(e), където е е агрегацията и се определя като $e = x_a w_a + x_b w_b$.

Пример

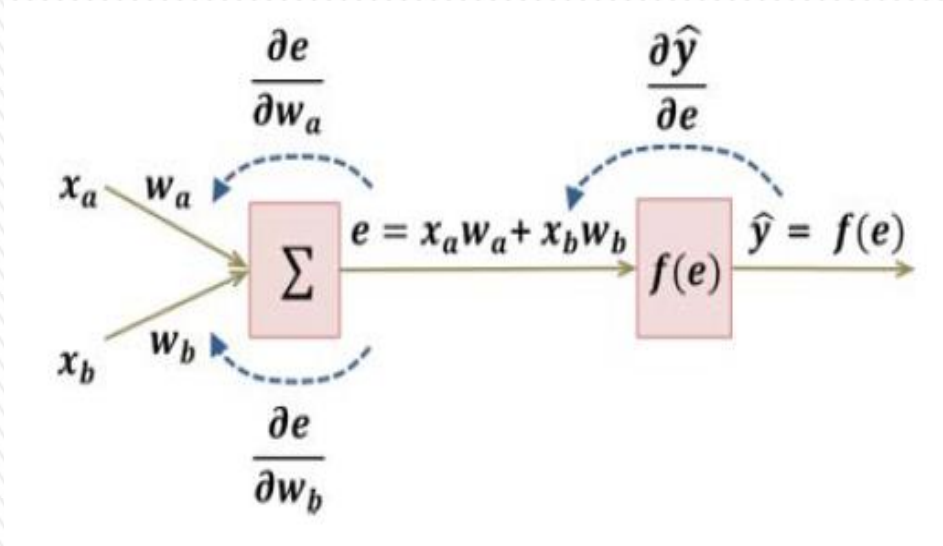


За да се минимизира грешката на ˆy, трябва да се изчисли нейната производна по отношение на e.

За праволинейна функция на активиране тогава имаме:

$$f'(e) = \frac{\partial \hat{y}}{\partial e}.$$

Пример

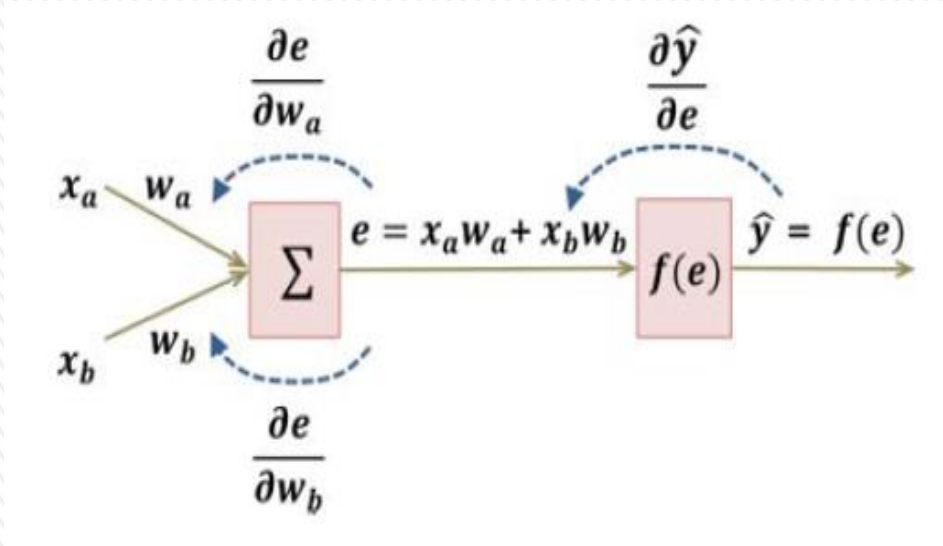


Частичните производни на ˆy по отношение на w_a и w_b са както следва:

$$\frac{\partial \hat{y}}{\partial w_a} = \frac{\partial (x_a w_a + x_b w_b)}{\partial w_a} = x_a$$

$$\frac{\partial \hat{y}}{\partial w_b} = \frac{\partial (x_a w_a + x_b w_b)}{\partial w_b} = x_b.$$

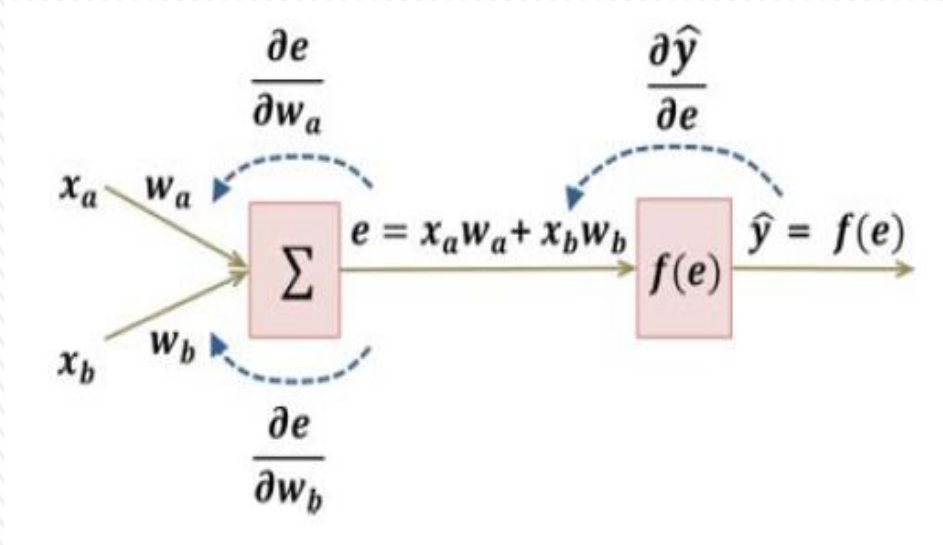
Пример



За реалната стойност на y при този неврон, грешка E се определя като:

$$E = \frac{1}{2}(y - \hat{y})^2.$$

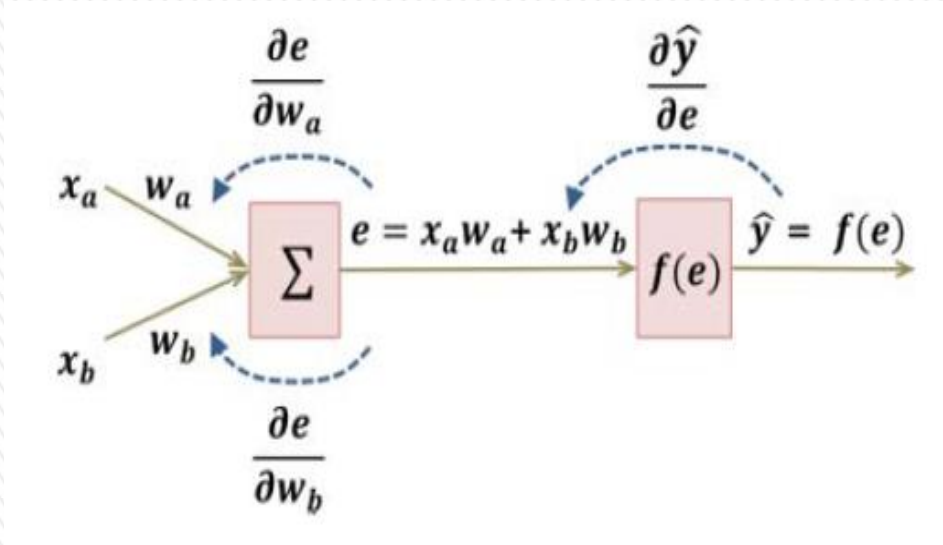
Пример



Частната производна на E по отношение на w_a e:

$$\frac{\partial E}{\partial w_a} = -(y - \hat{y})\frac{\partial \hat{y}}{\partial w_a} = -(y - \hat{y})\frac{\partial \hat{y}}{\partial e}\frac{\partial e}{\partial w_a}.$$

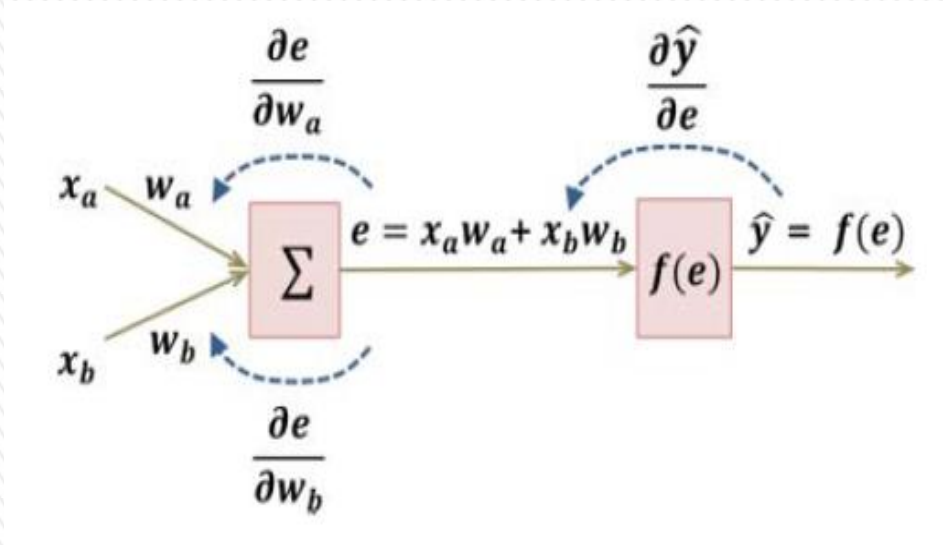
Пример



Нека дефинираме $\delta = -(y - \hat{y})f'(e)$ и го включим в горното уравнение за грешката:

$$\frac{\partial E}{\partial w_a} = \delta \frac{\partial e}{\partial w_a} = \delta x_i.$$

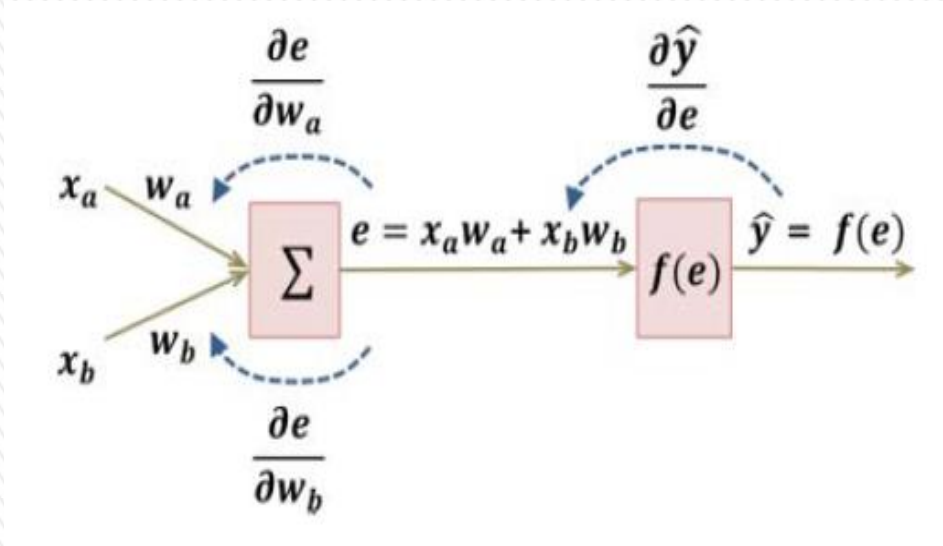
Пример



По подобен начин, частната производна на E по отношение на w_b e:

$$\frac{\partial E}{\partial w_b} = -(y - \hat{y})\frac{\partial \hat{y}}{\partial w_b} = -(y - \hat{y})\frac{\partial \hat{y}}{\partial e}\frac{\partial e}{\partial w_b}.$$

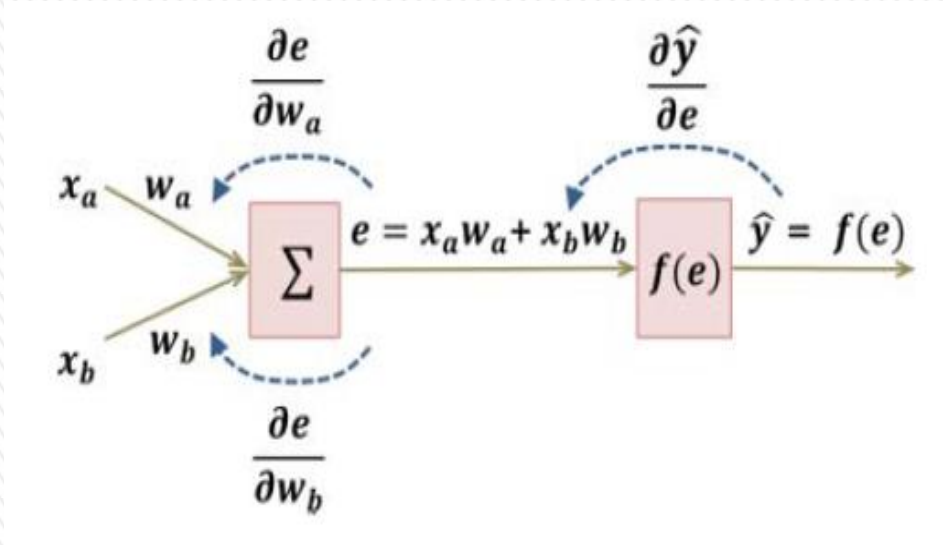
Пример



Отново, използвайки $\delta = -(y - \hat{y})f'(e)$ имаме:

$$\frac{\partial E}{\partial w_b} = \delta \frac{\partial e}{\partial w_b} = \delta x_j.$$

Пример



Използвайки стойностите от уравненията, актуализирането на теглата за скорост на обучение от α ще даде:

$$w_a = w_a - \alpha \frac{\partial E}{\partial w_a}$$

$$w_b = w_b - \alpha \frac{\partial E}{\partial w_b},$$

където α е реално число, определящо скоростта на модификация на тежестите по време на обучение.


Благодаря за вниманието!