Системи, основани на знания - зимен семестър,
2020/2021 учебна година

Тема 16:
Невронни мрежи

Същност на невронните мрежи

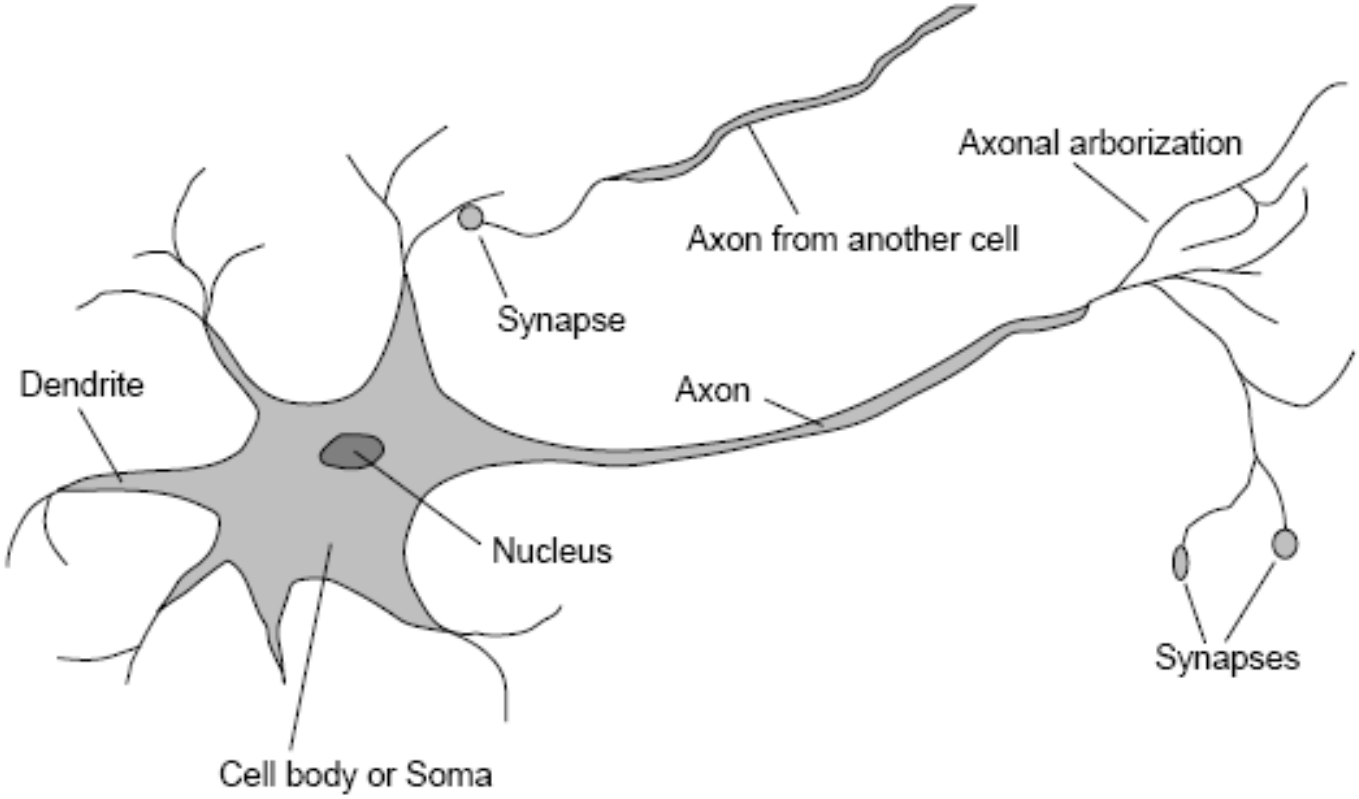
Невронните мрежи (НМ) и по-точно, изкуствените НМ са най-типичният и най-развитият представител на т. нар. *конекционистки подход* в Изкуствения интелект. Основните градивни елементи на (изкуствените) НМ са силно опростени модели на биологичните неврони, от които е съставен човешкият мозък.

Основни компоненти на биологичните неврони:

- *дендрити* – разклонени структури, които осигуряват сензорен вход към тялото на неврона (входни устройства на неврона);
- *тяло на неврона* – сумира мембранните потенциали между синапсите и дендритите и изпраща по аксона активиращо напрежение с определен размер;
- *аксон* – провежда електрическия сигнал от тялото на неврона до неговите синапси;

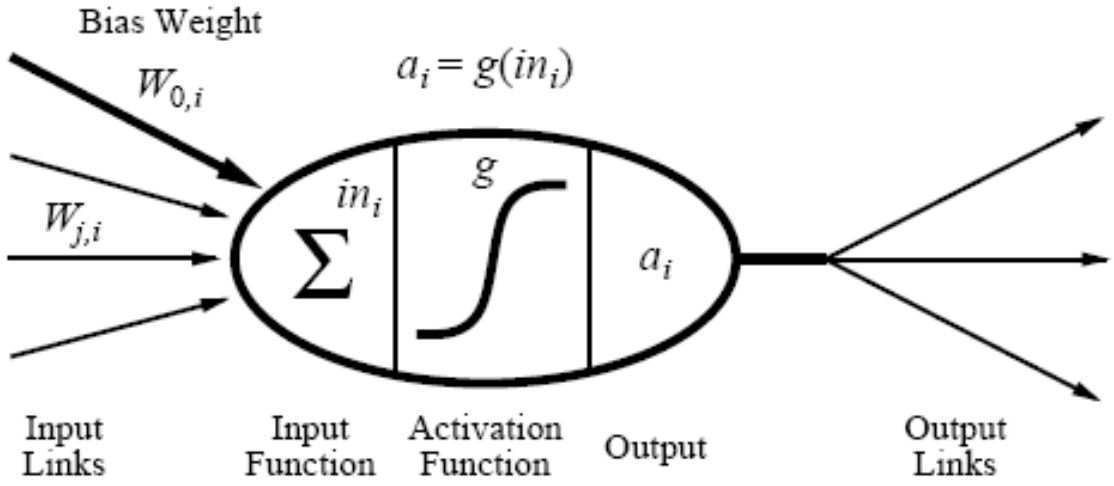
- *синапси* – трансформират активиращото напрежение, получено по аксона, в електрически сигнали (импулси), които възбуждат или подтискат активността на съседните неврони (изходни устройства на неврона).

Отделните неврони са свързани помежду си (дендритите на всеки неврон получават „входна информация“ от синапсите на други неврони и т.н.).


Axonal arborization
Axon from another cell
Synapse
Dendrite
Axon
Nucleus
Synapses
Cell body or Soma

Всяка изкуствена НМ е система от обработващи елементи (processing units), които са силно опростени модели на биологичните неврони и затова често условно се наричат (изкуствени) неврони. Обработващите елементи са свързани помежду си с връзки с определени тегла. Всеки обработващ елемент преобразува входните стойности (входните сигнали), които получава, и формира съответна изходна стойност (изходен сигнал), която разпространява към елементите, с които е свързан.

Това преобразуване се извършва на две стъпки. Най-напред се формира сумарният входен сигнал за дадения елемент, като за целта отделните входни стойности (сигнали) се умножават по теглата на връзките, по които се получават, и резултатите се събират. След това обработващият елемент използва специфичната за мрежата *активационна функция* (функция на изхода), която трансформира получения сумарен вход в изхода (изходния сигнал) на този елемент.


Bias Weight
$W_{0,i}$
$a_i = g(in_i)$
$W_{j,i}$
in_i
Σ
g
a_i
Input Links
Input Function
Activation Function
Output
Output Links

Основни типове модели на невронни мрежи

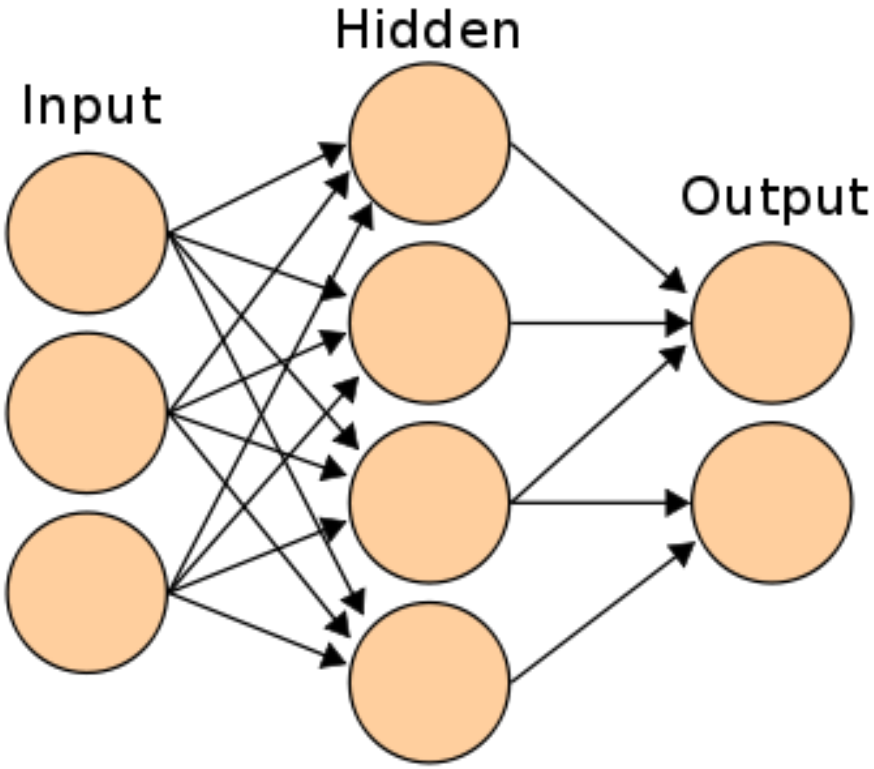
Специалистите определят пет основни характеристики на НМ, които обикновено се използват за класификационни признаци при определянето на типовете модели на НМ: *топологията на мрежата*, *параметрите на обработващите елементи*, *типът на входните и изходните стойности*, *методът за обучение на мрежата* и *използваното обучаващо правило*.

Според топологията на мрежата

Най-често срещани са НМ, съставени от няколко обособени (последователни) слоя от елементи, при които елементите от най-ниския слой играят ролята на входни устройства на мрежата (те възприемат сигнали от външната среда), а елементите от най-горния слой играят ролята на изходни устройства на мрежата (те извеждат резултата от работата на мрежата, който се получава на базата на входните сигнали и теглата на връзките между елементите).

Обикновено при тези НМ връзките са еднопосочни и свързват елементите от един слой с елементи на слоя, разположен непосредствено над него.

В зависимост от броя на слоевете в мрежата се говори за *двуслойни НМ* (при тях има само един входен и един изходен слой и липсват т. нар. *вътрешни* или *скрити слоеве*) и *многослойни НМ* (при тях има поне един скрит слой).



Многослойна (трислойна) НМ

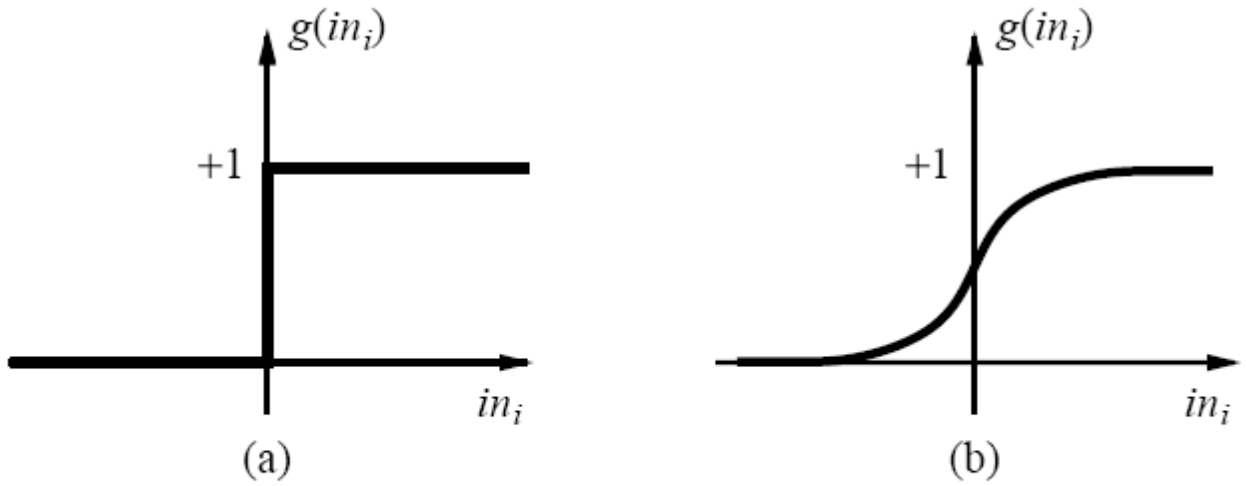
Според параметрите на елементите

Тук от значение са посоката и теглата на връзките, видът на активационната функция и др.

Както беше посочено, при повечето НМ връзките са еднопосочни и са ориентирани от елементите от даден слой към елементите от слоя, разположен непосредствено над него (това са т. нар. *прави мрежи*). Съществуват обаче и т. нар. *рекурентни мрежи* (такива са например мрежите на Хопфийлд), при които всеки елемент е свързан с двупосочни връзки с всички свои съседи. При тези мрежи понятието слой до голяма степен губи своя смисъл.

По отношение на теглата на връзките се говори за т. нар. *възбуждащи връзки* (връзки с положителни тегла) и *подтискащи връзки* (връзки с отрицателни тегла).

Най-често срещаните типове активационни функции са *стъпаловидна функция* (при персептрона) и *сигмоид* (при НМ с обратно разпространение на грешката).



(a) is a step function or threshold function

(b) is a sigmoid function $1/(1+e^{-x})$

Според типа на входните и изходните стойности

Входните стойности на мрежата (т.е. сигналите, които получават елементите от входния слой) могат да бъдат *двоични* (0 или 1) или *аналогови* (реални числа). Същото се отнася и за изходните стойности на мрежата (стойностите/сигналите, които елементите от изходния слой извеждат към „околната среда“). В случай, че изходните стойности на мрежата трябва да бъдат двоични, се налагат допълнителни изисквания към вида на активационната функция.

Според обучаващото правило

Най-често НМ се използват за решаване на задачи, свързани с разпознаване (класификация). При това създаването на НМ, предназначена за решаване на дадена конкретна задача, обикновено се извършва по следната обща схема. Най-напред се определя топологията на мрежата, т.е. броят на слоевете и броят на елементите във всеки слой. Броят на елементите от входния слой се определя от размерността на входните данни, а броят на елементите от изходния слой – от броя на разпознаваните класове. Броят и размерите на скритите слоеве се определят итеративно в зависимост от предметната област и конкретната задача.

След определянето на топологията на мрежата се преминава към нейното обучаване, т.е. към определянето на подходящи стойности на теглата на връзките между елементите. За целта най-често отначало се задават случайни стойности на търсените тегла, след което те итеративно се променят (уточняват) по такъв начин, че на следващата стъпка новият изход на мрежата да бъде по-добър от стария. Правилото, по което се променят теглата на връзките, се нарича *обучаващо правило* (обучаващ алгоритъм) на мрежата.

Примери за по-известни обучаващи правила: правило за обучение на персептрона (алгоритъм за обучение с фиксирано нарастване), алгоритъм за обучение с обратно разпространение на грешката, правило на състезателното обучение.

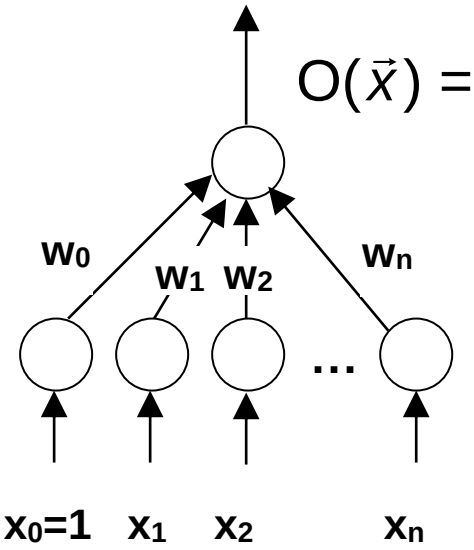
Според метода на обучение

Съществуват два основни типа обучение на НМ – *обучение с учител* и *обучение без учител*. При обучението с учител се следи разликата между получения и очаквания изход на мрежата (учителят задава очаквания изход на мрежата) и итеративно се извършват корекции на теглата на връзките съгласно избраното обучаващо правило. При обучението без учител липсват данни за правилния изход на мрежата. Теглата на връзките се настройват така, че представянето на данните в мрежата да е най-добро съгласно използвания (зададения) критерий за качеството на представянето.

Популярни модели на невронни мрежи и алгоритми за обучение

Персептрон

Персептронът (по-точно, двуслойният персептрон) е двуслойна НМ с един елемент в изходния слой, който често се нарича сумиращ процесор на персептрона.



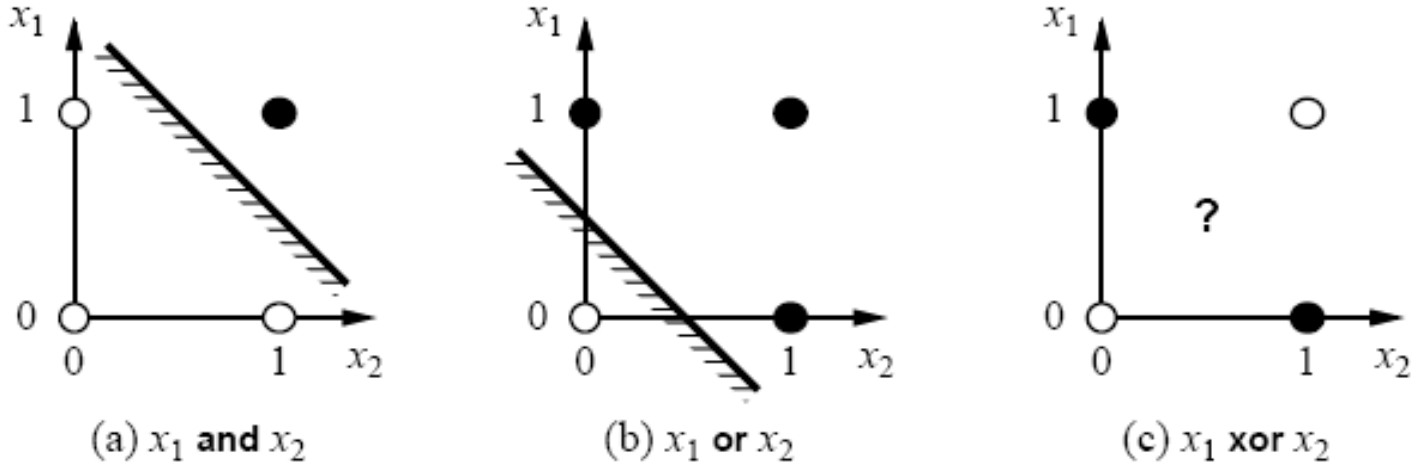
$$O(\vec{x}) = \begin{cases} 1, \text{ ако } g(\vec{x})>0 \\ 0, \text{ ako } g(\vec{x})<0 \\ \text{Резултатът е неопределен, ако } g(\vec{x})=0 \end{cases}$$

Тук $g(\vec{x}) = w_0 + \sum_{i=1}^{n} w_i x_i = \vec{w}\vec{x}$ ($\vec{w}$ - теглови вектор,

$\vec{x}$ - входен вектор)

Резултатът от работата на персептрона е решаването на задача за разпознаване при дадени два класа, разделими с хиперравнина (в двумерния случай - разделими с права линия, т.е. линейно разделими).

Примери



(a) x_1 **and** x_2 (b) x_1 **or** x_2 (c) x_1 **xor** x_2

Алгоритъм за обучение на персептрона – т. нар. *алгоритъм за обучение с фиксирано нарастване*. Доказана е теорема за сходимост, според която ако съществува множество от тегла, определящи хиперравнина, която разделя двата класа, то с помощта на този алгоритъм могат да бъдат намерени подходящи стойности на търсените тегла на връзките между елементите от входния и изходния слой.

Формулировка на алгоритъма за обучение на персептрона

Дадено: множество от обучаващи примери за задача за разпознаване (класификация) с n входни характеристики (x_1, … , x_n) и два изходни класа.

Търси се: множество от тегла (w_0, w_1, … , w_n), които са такива, че персептронът дава стойност 1 точно тогава, когато входните данни съответстват на обект, принадлежащ на първия клас.

Изчислителна схема:

1. Създава се персептрон с $n+1$ входни елемента и $n+1$ тегла, където допълнителният входен елемент x_0 винаги има стойност 1.
2. Инициализират се със случайни реални стойности теглата (w_0, w_1, … , w_n).
3. При текущите стойности на теглата w_i се класифицират примерите от обучаващото множество, след което от тях се подбират само тези, които са класифицирани неправилно.
4. Ако всички обучаващи примери са класифицирани правилно, като резултат от работата на алгоритъма се извеждат текущите стойности на теглата w_i и ***край***.

5. В противен случай се пресмята векторът $\vec{s}$ като сума на неправилно класифицираните обучаващи вектори $\vec{x}$. При формирането на сумата $\vec{s}$ в нея с положителен знак участват всички вектори $\vec{x}$, за които персептронът неправилно е дал резултат 0, а с отрицателен знак – тези, за които персептронът неправилно е дал резултат 1. Така полученият вектор $\vec{s}$ се умножава с предварително избран скаларен коефициент η. Стойността на η ($0<\eta<1$) определя скоростта на сходимост на алгоритъма и изборът ѝ зависи от спецификата на решаваната задача.

6. Модифицират се теглата (w_0, w_1, … , w_n), като към тях се прибавят съответните елементи на вектора $\vec{s}$. Преминава се към т. 3.

Забележка. Нека е дадено множество от обекти, които подлежат на разпознаване/класификация. Предполага се, че всеки обект се характеризира с набор от съществени признаци. На базата на използваните признаци се определя *признаковото пространство* (*пространството на признаците*). Всеки обект е точка в това пространство. Тогава задачата за дефиниране на класовете се свежда до това за всеки клас C_i да се намери функция $g_i(\vec{x})$, която е такава, че $\vec{x} \in C_i$ тогава и само тогава, когато $g_i(\vec{x}) > g_j(\vec{x})$ за всяко $j \neq i$.

При известни функции $\{g_i\}$ задачата за класификацията на даден обект $\vec{x}$ се свежда до намиране на това k, за което $g_k(\vec{x})>g_j(\vec{x})$ за всяко $j\neq k$ (тогава $\vec{x}\in C_k$). При наличие само на два класа горната задача се свежда до намиране или използване на т. нар. *разделящо правило* (*разделяща функция*) g със свойството: $\vec{x}\in C_1$, когато $g(\vec{x})>0$, и $\vec{x}\in C_2$, когато $g(\vec{x})<0$. В този случай точките, за които $g(\vec{x})=0$, образуват границата между класовете.

Невронни мрежи с обратно разпространение на грешката

Това са НМ с поне един скрит слой, при които съществуват връзки от всеки елемент от даден слой към всеки елемент от непосредствено следващия слой.

Активационната функция на този тип мрежи има вида $output = \frac{1}{1+e^{-sum}}$, където *sum* е сумата от всички входни стойности, умножени с теглата на съответните връзки.

Алгоритъм за обучение с обратно разпространение на грешката: включва множество *епохи*, в рамките на всяка от които последователно се разглеждат всички примери от обучаващото множество. За всеки обучаващ пример се извършват т. нар. *прав пас* (разпространение на активационните стойности) и *обратен пас* (анализ на грешките на изхода и разпространение на корекциите на теглата на връзките между елементите от различните слоеве).

Следва описание на алгоритъма за случая на трислойна невронна мрежа (НМ с точно един скрит слой).

Дадено: множество от обучаващи примери (наредени двойки $(\vec{x},\vec{y})$ от входни и съответни изходни стойности на мрежата).

Търсят се: подходящи стойности $\overrightarrow{w_1}$ и $\overrightarrow{w_2}$ на теглата на връзките в трислойна НМ, която по дадени входни стойности от обучаващото множество връща съответните изходни стойности от същото множество (т.е. работи коректно върху даденото множество от обучаващи примери).

Изчислителна схема:

1. Нека A е броят на елементите от входния слой (съответен на дължината на входните вектори от обучаващото множество) и C е броят на елементите от изходния слой (съответен на дължината на изходните вектори от обучаващото множество). Избира се подходяща стойност на B, равна на броя на елементите от скрития слой. Има многобройни резултати, които могат да се използват при експериментиране в процеса на определянето на B. Въвеждаме следните означения:
   x_i – стойностите (активационните равнища) на елементите от входния слой (при това се полага $x_0 = 1$);
   h_i – стойностите (активационните равнища) на елементите от скрития слой (при това се полага $h_0 = 1$);

o_i – стойностите (активационните равнища) на елементите от изходния слой;
$w_1(i,j)$ – теглата на връзките между елементите от входния и скрития слой (i – индекс на елемента от входния слой, j – индекс на елемента от скрития слой);
$w_2(i,j)$ – теглата на връзките между елементите от скрития и изходния слой (i – индекс на елемента от скрития слой, j – индекс на елемента от изходния слой).

2. Инициализират се теглата на връзките в мрежата. Всяко тегло получава случайна стойност в интервала (-0.1,0.1).
3. Инициализират се стойностите (активационните равнища) на допълнителните елементи: x_0 = 1.0, h_0 = 1.0. Тези стойности не се променят на следващите стъпки от изпълнението на алгоритъма.
4. Избира се обучаващ пример, т.е. двойка ($\vec{x}$,$\vec{y}$) от входен и съответен изходен вектор от обучаващото множество.

5. Разпространяват се активационните стойности от входния към скрития слой, като за целта се използва активационната функция на мрежата:
   $h_j = \frac{1}{1+\exp(-\sum_{i=0}^{A} w_1(i,j)x_i)}$ за j= 1,2, ... ,B.
6. Разпространяват се активационните стойности от скрития към изходния слой:
   $o_j = \frac{1}{1+\exp(-\sum_{i=0}^{B} w_2(i,j)h_i)}$ за j= 1,2, ... ,C.
7. Пресмятат се грешките, получени при елементите от изходния слой. Нека означим тези грешки с $\delta_2(j)$. Всяка от тях се пресмята с помощта на реалния изход на мрежата o_j и правилния изход от обучаващия пример y_j:
   $\delta_2(j) = o_j(1 - o_j)(y_j - o_j)$ за j = 1,2, … ,C.

8. Пресмятат се грешките на елементите от скрития слой.Нека означим тези грешки с $\delta_1(j)$:
   $\delta_1(j) = h_j(1 - h_j)\sum_{i=1}^{C} \delta_2(i) w_2(j,i)$ за $j = 1,2, \ldots, B$.
9. Уточняват се стойностите на теглата на връзките между елементите от скрития и изходния слой:
   $w_2(i,j) = w_2(i,j) + \Delta w_2(i,j)$,
   $\Delta w_2(i,j) = \eta \delta_2(j) h_i$ за $i = 0, \ldots, B$ и $j = 1, \ldots, C$.
   Тук коефициентът η има същия смисъл, както и при алгоритъма за обучение на персептрона.
10. Уточняват се стойностите на теглата на връзките между елементите от входния и скрития слой:
    $w_1(i,j) = w_1(i,j) + \Delta w_1(i,j)$,
    $\Delta w_1(i,j) = \eta \delta_1(j) x_i$ за $i = 0, \ldots, A$ и $j = 1, \ldots, B$.

11. Преминава се към стъпка 4 и се повтарят действията от стъпки 4 – 10 за всички останали обучаващи примери.

Така завършва една *епоха* от работата на алгоритъма. Повтарят се толкова епохи, колкото е необходимо за достигане на коректен изход на мрежата за всички обучаващи примери.

Алгоритъмът за обучение с обратно разпространение на грешката има *ниска скорост на сходимост*, която води до ниска скорост на обучение на невронната мрежа и необходимост от *голям брой обучаващи примери*.

НМ с обратно разпространение на грешката се използват широко поради голямата им мощност.

Обучение на невронни мрежи без учител (самообучение на невронни мрежи)

При обучението без учител (т.е. при самообучението) се предполага, че не съществува обратна връзка със средата, в която работи мрежата, т.е. няма кой да определи какъв би трябвало да бъде изходът на мрежата и дали формираният от нея изход е коректен. Мрежата трябва сама да открие образци, характеристики, регулярности, корелации или категории във входните данни и да ги кодира на изхода. Затова изкуствените неврони и връзките между тях трябва да притежават (да демонстрират) някаква степен на ***самоорганизация***.

В резултат на самообучението може да се получи нещо полезно само когато съществува ***изобилие от входни данни***. Без изобилие не може да се намерят някакви характеристики на данните, тъй като в тях има и случайни шумове. В този случай ***изобилието дава знание***.

Типът на образеца, който самообучаващата се мрежа открива във входните данни, зависи от ***архитектурата ѝ***. Обикновено самообучаващите се мрежи имат проста архитектура – най-често те са прави и включват един входен и един изходен слой. При това обикновено изходите са много по-малко от входовете.

Някои възможни резултати от работата на самообучаваща се невронна мрежа:

Клъстеризация. Множество от двоични изходи, само един от които е активен в определен момент от времето, могат да определят към коя от няколко категории принадлежи входният образец. Всеки клъстер от подобни или близки образци ще бъде класифициран като един изходен клас.

Кодиране. Изходът може да бъде кодирана версия на входа в по-малко битове, поддържайки толкова необходима информация, колкото е възможно. Това се използва за компресиране на данни преди подаването им по канал с ограничен обхват, предполагайки, че може да се конструира и обратна декодираща мрежа.

В повечето случаи архитектурите и обучаващите правила се основават на интуитивни предположения. В някои случаи се използват и оптимизационни подходи (цели се максимална икономичност на представянето или максимизиране на някакво количество, например на съдържанието на информацията, или минимизиране на изменението на изхода).

Един от най-често използваните методи (подходи) за самообучение е т. нар. ***състезателно обучение***. При него ***само един изходен неврон или само един неврон от определена група е активен***, т.е. има ненулева активност (активационно ниво, стойност) в даден момент от времето. Изходните неврони се състезават да бъдат активни и затова често се наричат ***„печелившият-взема-всичко“*** неврони или ***„изпреварващи-всички“*** елементи (възли).

Целта и тук е да се клъстеризират или категоризират входните данни. Подобни входове би трябвало да бъдат класифицирани в една и съща категория и затова би трябвало да активират един и същ изходен неврон. Класовете (клъстерите) трябва да бъдат открити от самата мрежа на основата на корелации между входните данни.

Стандартно състезателно обучение

Мрежите, които използват стандартното състезателно обучение, имат ***един входен и един изходен слой***. Всеки от изходните възли O_i е свързан с всеки от входните възли ξ_j чрез възбуждаща връзка с тегло $w_{ij} \geq 0$. Ще разглеждаме само мрежи с ***бинарен вход и изход*** (с входни и изходни стойности от множеството {0,1}). Само един от изходните неврони, наречен ***победител***, може да бъде активен (да има ненулева стойност) в даден момент. Обикновено това е възелът с най-голяма стойност (активност)

$h_i = \sum_j w_{ij}\xi_j = \overrightarrow{w_i}\overrightarrow{\xi}$ за дадения входен вектор $\overrightarrow{\xi}$.

Затова неравенството

(1) $\overrightarrow{w_{i*}}\overrightarrow{\xi} \geq \overrightarrow{w_i}\overrightarrow{\xi}$ за всяко *i*

дефинира побеждаващия неврон с $O_{i*} = 1$.

Като правило се изисква теглата за всеки изходен възел да са нормирани:

$|\overrightarrow{w_i}| = 1$ за всяко *i*.

В такъв случай за даден входен вектор $\vec{\xi}$ побеждаващият изходен неврон *i** с O_{i*} = 1 се дефинира посредством неравенството

(2) $|\overrightarrow{w_{i*}}-\vec{\xi}| \leq |\overrightarrow{w_i}-\vec{\xi}|$ за всяко *i*.

Една мрежа от тип „печелившият-взема-всичко“ реализира класификатор на образци, като използва критерия (1) или (2). ***Задачата за обучението ѝ*** е свързана с ***намиране на тегловите вектори*** $\overrightarrow{w_i}$ ***при изискването мрежата да намира по подходящ начин клъстери във входните данни***.

Няма принципно значение начинът, по който се реализира мрежа от тип „печелившият-взема-всичко“. При компютърна симулация винаги може да се осъществи търсене на максималното h_i. Често срещана (и по-естествена в определен смисъл) е и ситуацията, при която изходните неврони се състезават помежду си с цел излъчване на победител чрез ***странично подтискане***: всеки неврон подтиска другите чрез връзки с отрицателни тегла и евентуално се самовъзбужда чрез връзка с положително тегло. За целта страничните тегла и нелинейната активационна функция трябва да бъдат подбрани коректно – така, че да е сигурно, че ще бъде избран точно един изход и колебанията ще бъдат избегнати.

В такъв случай най-общо алгоритъмът на стандартното състезателно обучение изглежда по следния начин:

1. Присвояват се малки случайни стойности на теглата. Важно в този момент е да няма симетрия (т.е. теглата да са напълно различни).
2. Избира се входен вектор от обучаващото множество.
3. Пресмята се началната стойност (активност, активационно ниво) за всеки от изходните неврони.
4. Изходните неврони се състезават, докато само един от тях остане активен.

5. Увеличават се теглата на връзките между активния изходен неврон и активните входни неврони и се намаляват теглата на връзките между активния изходен неврон и неактивните входни неврони така, че векторът от тези тегла да остане нормиран.
6. Стъпки 2 – 5 се повтарят за всички входни вектори за много епохи.