A causa de les limitacions del perceptró, l’any 1969, Marvin Lee Minsky i Seymour Aubrey Papert van publicar un llibre anomenat *Perceptrons: an introduction to computational geometry,* en el qual parlaven de les severes limitacions que tenia el perceptró. L’efecte del llibre va ser tant gran, que la recerca en el tema de les xarxes neuronals es va aturar completament durant casi 10 anys.

Després d’una sèrie de retallades de pressupost i poc avanç en el tema, vam arribar a principis del segle XXI, que amb grans bases de dades i ordinadors molt més potents, es van poder fer molts més avenços. Finalment, van sorgir les anomenades *Multylayer perceptron*, o Xarxes neuronals artificials.

**Xarxes neuronals artificials (XNA)**

1. **Estructura de les XNA**

Les xarxes neuronals artificials, estan formades per perceptrons connectats entre si, organitzats en tres capes: entrada, capa oculta i sortida.



La primera capa, la capa d’entrada, és la que s’encarrega de rebre les dades que haurà de processar; un cop les ha rebut, les envia a la capa amagada, que és l’encarregada de processar aquestes dades i enviar-les a l’última capa: la sortida, que rep les dades de la capa amagada, i ens retorna un resultat.

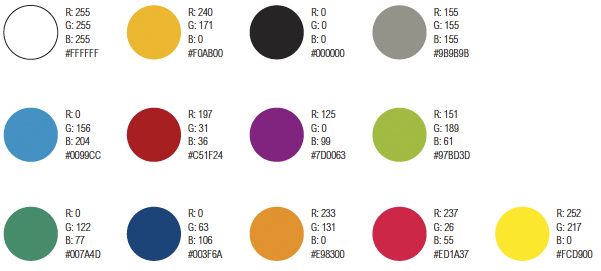
El nombre de neurones de l’entrada i la sortida, dependrà de perquè volem utilitzar la xarxa neuronal. Per exemple: si la volem utilitzar per agafar dos nombres, sumar-los i que ens retorni el resultat, aquesta xarxa tindrà dues entrades (els dos nombres que volem sumar) i una sortida (el resultat); en canvi, si volem que sumi tres nombres, tindrà tres neurones d’entrada.

1. **Funcionament de les XNA**

***Feedforward***

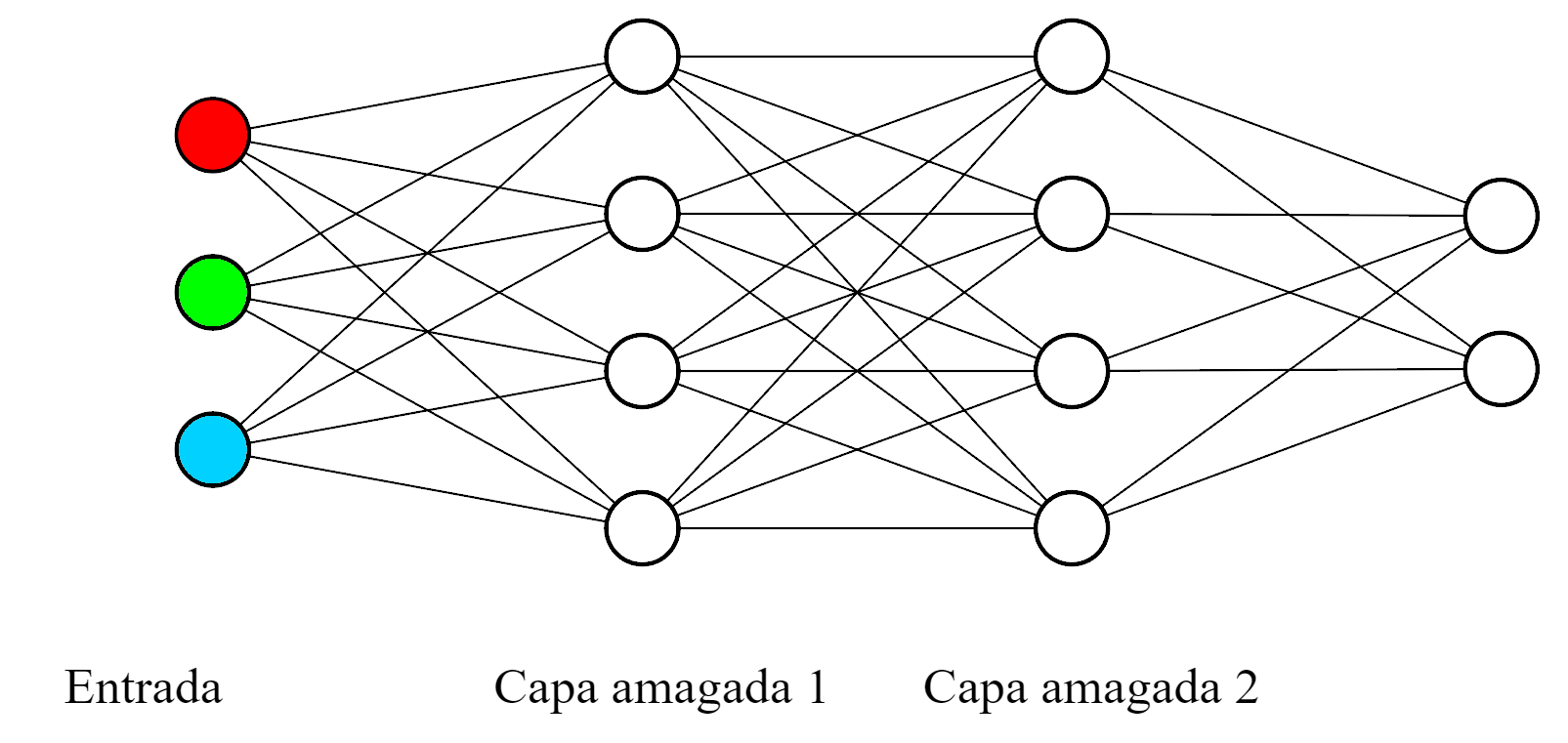
*Feedforward* és l’algoritme que utilitzarem perquè la nostra xarxa neuronal ens retorni un resultat amb unes dades que l’hi introduïm.

A aquesta xarxa neuronal artificial, l’hi introduirem un color en valor *rgb*, i ens dirà quina és la possibilitat de que aquell color sigui clar o sigui fosc. Per tant tindrà tres neurones d’entrada, la primera per la quantitat de vermell del color, la segona per la quantitat de verd i la tercera per la quantitat de blau.



*En aquesta imatge es poden apreciar diferents colors amb el seu valor rgb, cada un dels valors r, g o b té un número d’entre 0 i 255; i els tres combinats poden formar tots els colors de l’espectre visible.*

De neurones de sortida, en tindrà dues; la primera representarà la possibilitat de 0 a 1 de que el color sigui clar, i la segona representarà la possibilitat de que el color sigui fosc. La suma de les dues possibilitats sempre donarà 1. En la capa amagada, tindrà dues columnes de quatre neurones cada una. Sabent ja el nombre de columnes i de neurones a cada una d’elles, podem fer el diagrama, que ens ajudarà a entendre millor el seu funcionament.

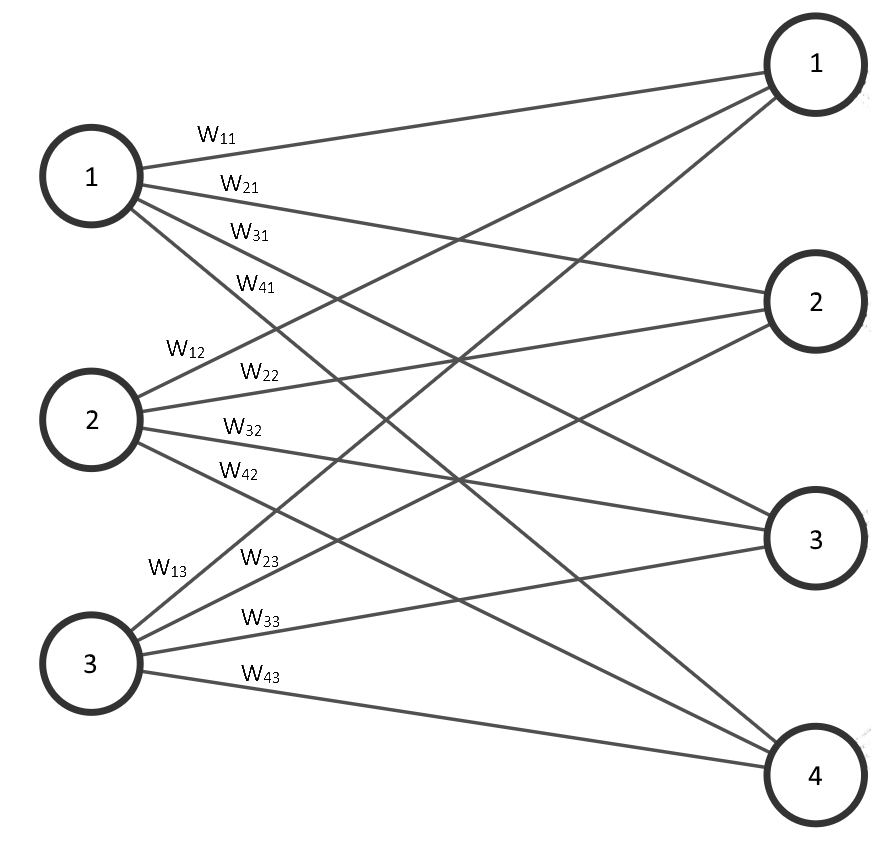


Ara que ja tenim definida l’estructura de la nostra xarxa neuronal, hem de tenir clar com emmagatzemarem les dades, tant les d’entrada com els valors de cada un dels pesos, per facilitar i optimitzar el funcionament.

Per emmagatzemar les dades, utilitzarem matrius. En total, per aquesta xarxa neuronal, utilitzarem 7 matrius. De moment ens fixarem en les dues primeres. La primera emmagatzemarà el valor de les senyals d’entrada, i la segona emmagatzemarà el valor de les connexions de la primera columna a la segona. Per organitzar-ho millor, donarem números a les neurones de cada columna i als pesos de la següent manera:

* Les neurones s’enumeraran per columnes, la primera neurona de cada columna serà la nº 1, la de sota serà la nº 2 i així fins a arribar al final de la columna.
* Cada un dels pesos tindrà dos subíndex: wab, el primer farà referència al nombre de neurona al que va i el segon farà referència al nombre de neurona del que ve.

Seguint aquestes normes d’enumeració, les dues primeres columnes de la xarxa neuronal haurien de quedar numerades d’aquesta manera:



I ara que ja ho tenim tot enumerat, podem crear les matrius. La primera, la de les senyals d’entrada, quedaria de la següent manera: . I la matriu dels pesos de la següent manera: , seguint la norma de que el primer número del subíndex es refereix a la fila de la matriu i el segon nombre a la columna.

Ara que ja tenim les matrius preparades, ja podem introduir-li dades a la xarxa i fer que ens torni un resultat. Per provar la nostra xarxa neuronal, farem que ens digui si un color groguenc, de valors r: 252, g:217 i b: 0 (veure a la imatge nº jo), és clar o fosc. Aleshores, tenim que les tres dades d’entrada són: 252, 217 i 0. Perquè la nostra xarxa neuronal treballi bé, primer de tot farem que normalitzi els valors, és a dir, que els passi a un rang d’entre 0 i 1; i això ho farem dividint els tres valors d’entrada entre 255. En aquest punt, ja tenim la matriu de les senyals d’entrada omplerta, amb els següents valors: .

A la matriu dels pesos, definirem els valors de manera aleatòria. I d’aquesta manera, ens queda així: .

Ara ja podem fer el mateix que vam fer amb el perceptró: primer de tot, hem de calcular un resultat inicial, i després el passarem per una funció d’activació. Per calcular el resultat inicial, multiplicarem la matriu dels pesos per la matriu dels nombres d’entrada. Si ho hem fet tot bé, la matriu resultant, hauria de ser un vector de quatre nombres, que correspondran als resultats per cada una de les neurones de la segona columna.

x = = .

Ara que ja tenim un resultat inicial, l’haurem de passar per una funció d’activació. Aquesta funció d’activació, anomenada *sigmoid*, ens retornarà un valor entre 0 i 1 per qualsevol nombre que l’hi introduïm. La fórmula de la funció *sigmoid* és: , tenint en compte que z és el resultat inicial, i la seva gràfica és:



Ara hem de repetir la mateixa operació dos cops més: un per arribar a la segona columna de la capa oculta i un segon per arribar a les senyals de sortida. Cal tenir en compte que en les dues operacions, la matriu dels pesos varia respecte la primera i sempre té valors aleatoris.

Si recordem, teníem dues neurones a la capa de sortida, i volíem que cada neurona ens retornés una possibilitat del 0 al 1, tenint en compte que les dues possibilitats sumades havien de donar 1. Això ho farem canviant la funció d’activació, però només per aquesta operació. La funció d’activació que utilitzarem s’anomena *softmax.*

***Softmax function***

En la nostra xarxa neuronal, la capa de senyal de sortida ha calculat un resultat inicial dels valors que ha rebut de la capa anterior, i aquests resultats inicils són: . Ara, la funció d’activació *softmax* transformarà aquests resultats a probabilitats, fent que la suma dels dos sigui 0. Si mirem a la funció veiem que . El que aquesta funció fa, per cada un dels resultats, és elevar el nombre *e* al resultat i dividir-ho entre la suma del nombre *e* elevat a cada un dels resultats. Per tant, en el nostre cas quedaria: resultat 1 = = 0,96, resultat 2 = = 0,04.

***Backpropagation***

Per entrenar la nostra xarxa neuronal, utilitzarem un mètode anomenat: “Aprenentatge supervisat”.

1. **Aprenentatge supervisat**

El procés d’entrenament de l’aprenentatge supervisat, es divideix en dues parts: el càlcul d’error i la rectificació dels pesos.

1. **Càlcul d’error**

El càlcul d’error de les XNA és més complicat al del perceptró, ja que aquestes tenen moltes més connexions i neurones. Per entendre ben bé que està passant al entrenar la XNA, ho farem començant per l’última columna i acabant a la primera, ja que és la manera de treballar de l’algoritme.

Per tant, començarem per la columna de la sortida que, ja que té dues neurones, haurem de calcular l’error per cada una d’elles. Per fer-ho utilitzarem la mateixa fórmula que amb el perceptró:

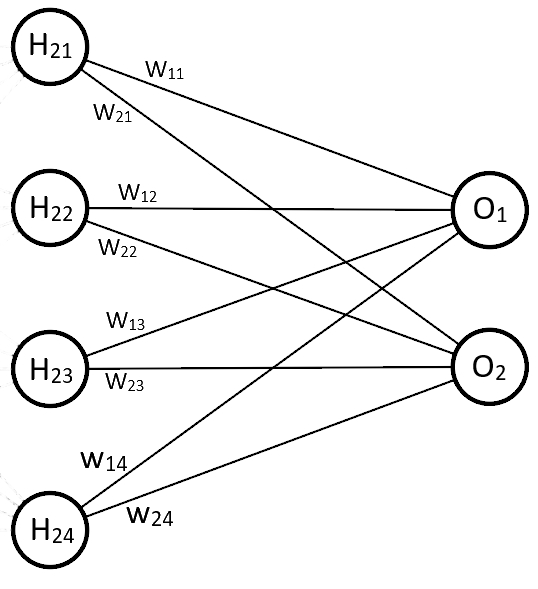
**error = resposta desitjada – resposta de la XNA**

Tenint això en compte, ho hem de calcular. Tant la resposta desitjada (que l’hi introduirem nosaltres) com la resposta de la XNA (que ha calculat anteriorment), són vectors amb dues dades. La resposta que ens ha calculat la XNA (a la qual anomenarem “*out*”) és *out* = i com que l’hi hem dit que calculi un color groguenc, el resultat hauria de ser , ja que es tracta d’un color clar. Per tant, per calcular l’error de les dues neurones de la capa de sortida, farem una simple resta de matrius:

eo = - = =

Hem anomenat aquest vector d’errors “eo” *e* referint-se a error i *o* referint-se a *output*, les senyals de sortida.

Ara que ja hem calculat l’error de les neurones de la capa de sortida, hem de calcular l’error de la columna anterior, la segona columna de la capa amagada:



El principal inconvenient de calcular l’error d’aquesta capa, és que cada una de les neurones té dues connexions, i cada connexió va a una neurona amb un error diferent; per tant, no podem utilitzar la mateixa fórmula que per la senyal de sortida.

En canvi, la fórmula que utilitzarem per calcular l’error de cada una de les neurones “H”, serà fer la suma dels pesos d’aquella neurona multiplicats per l’error de la neurona a la que van. Per tant l’error de la neurona H21, serà el pes w11 multiplicat per l’error e1 i el pes w21 multiplicat per l’error e2. Si ho mirem des d’un punt de vista matemàtic, podrem veure que s’assembla a una multiplicació de matrius, i les matrius que multipliquem són la dels pesos, transposada, i la de l’error de sortida.

w = , wT =

*eH = wT · eo = ·* =

La matriu resultant té quatre nombres, i cada un d’ells equival a l’error d’una de les neurones de la tercera columna. Per anomenar aquells errors, hem utilitzat la següent nomenclatura: eH2x, a on *e* es refereix a error, ja que són els valors de l’error; *H* es refereix a *hidden*, la capa oculta; el número 2 es refereix a que és l’error de la segona columna de la capa oculta i l’últim nombre, fa referència a la neurona.

Ara, hem de repetir aquest procés dos cops més: un per la primera columna de la capa oculta i una altre per la columna d’entrada. Quan haguem calculat cada un dels errors de cada neurona, haurem acabat amb quatre matrius d’errors, una per cada columna; i només quedarà variar els pesos.