Per entendre com funcionen les xarxes neuronals artificials, primer de tot ens fixarem en l’exemple més senzill: el perceptró. El perceptró és la xarxa neuronal artificial més simple, el qual consta només d’una neurona. Aquesta neurona rebrà unes senyals d’entrada, farà una càlculs i tornarà un resultat.



El nostre perceptró tindrà dues senyals d’entrada, anomenades x1 i x2, i cada una d’aquestes senyals estarà connectada a la neurona mitjançant una connexió. Aquesta connexió pot ser més forta o més dèbil; i això ho indicarem amb el pes (w11, w12), un número per cada connexió. Per facilitar les operacions més endavant, tant els valors de la senyal d’entrada com els valors dels pesos, els emmagatzemarem en matrius. Per fer-ho, seguirem les següents normes:

* Les senyals d’entrada s’emmagatzemaran en un vector
* Els pesos tindran dos subíndex, el primer farà referència al nombre de neurona al que es dirigeix, i el segon farà referència al nombre de neurona del que ve.
* Els subíndex dels pesos, fan referència a la posició en que han d’estar dins la matriu, el primer nombre referint-se a la fila i el segon nombre referint-se a la columna. D’aquesta manera el pes w11 estarà a la primera fila i primera columna de la matriu, i el pes w12 estarà a la primera fila i la segona columna.

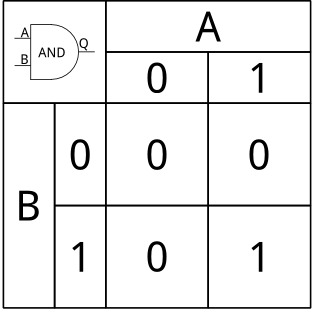
Entenent això, ja podem crear les matrius, que haurien de quedar de la següent manera:

Senyals d’entrada:

Pesos:

**Feedforward**

Ara que ja tenim l’estructura del perceptró, és hora de donar-li unes senyals d’entrada i fer que el nostre perceptró ens doni una resposta. Per poder veure si funciona o no, intentarem que el nostre perceptró resolgui portes lògiques, en concret la porta lògica AND.



Aquesta porta lògica només tornarà com a resposta un 1 si les dues senyals d’entrada són 1; en cas contrari tornarà 0.

Al iniciar el nostre perceptró, els pesos de les connexions es donaran de manera aleatòria, ja que no tenim manera de saber quins han de ser.

Fet això, li donem dues senyals d’entrada, per exemple: 0 i 1. Per calcular el resultat, el perceptró farà dues operacions: calcular un resultat inicial i convertir-lo mitjançant una **funció d’activació**.

Per calcular el resultat inicial (z), el que el perceptró fa és agafar les senyals d’entrada i multiplicar-les pel pes corresponent. Després es farà la suma de tots els resultats. Com que estem emmagatzemant els valors en matrius, una multiplicació de matrius ens donarà el mateix resultat. Per tant podem dir que el resultat inicial és igual a la matriu dels pesos multiplicada per la matriu de les senyals d’entrada.

***z = w · x***

x = 3 · 0 + 15 · 1 = 15

Al iniciar, el nostre perceptró ha assignat aleatòriament el valor 3 al pes 1 (w11) i el valor 15 al pes 2 (w12); per tant, el perceptró ens ha tornat -15 com a resultat inicial. Ara hem de normalitzar el resultat, és a dir, transformar el resultat a un rang de números que ens sigui útil. Com que estem treballant amb portes lògiques, el nostre rang de nombres adequat seria entre 0 i 1.

Per transformar aquest resultat al rang que necessitem ho farem amb una funció d’activació. La funció d’activació que nosaltres utilitzarem serà una anomenada *unit step*. Si nosaltres li introduïm qualsevol nombre a aquesta funció sempre ens en retornarà un entre 0 i 1. Aquesta funció, és una funció composta, de fórmula: .



Un cop el perceptró ja ens ha tornat un nombre entre 0 i 1, significa que ja ha fet els seus càlculs i ens ha donat una resposta per les dues senyals d’entrada que l’hi hem donat. Si recordem, les senyals d’entrada eren 0 i 1, i el resultat, ja passat per la funció d’activació, és 1. Com que nosaltres volíem que el perceptró resolgués la porta lògica AND, significa que aquest resultat està malament, ja que hauria de ser 0. Així que ara ens tocarà entrenar el perceptró perquè retorni el resultat que volem.

***Backpropagation***

El *backpropagation algorithm* o l’algoritme de propagació cap enrere, és una part del codi de la xarxa neuronal o del perceptró amb la tasca de modificar tots i cada un dels pesos de cada connexió per aconseguir la resposta desitjada. L’algoritme d’aprenentatge que utilitzarem pel perceptró s’anomena *supervised learning*.

1. **Aprenentatge supervisat**

L’algoritme d’aprenentatge supervisat, és un tipus d’algoritme que s’utilitza per entrenar el perceptró o la xarxa neuronal.

Per utilitzar aquest algoritme, necessites donar-li unes senyals d’entrada al perceptró de les quals tu ja en sàpigues la resposta. Per exemple, en el cas anterior, les senyals d’entrada eren 0 i 1, i la resposta esperada era 0; però el perceptró ha donat 1. Per tant, és hora d’entrenar el perceptró. Quan diem entrenar, ens referim a canviar els pesos per obtenir la resposta desitjada.

Abans de començar a canviar pesos, hem de veure com de malament ho ha fet el nostre perceptró, calculant l’error. Per calcular l’error direm que:

**error = resposta desitjada – resposta del perceptró**

Seguint aquesta fórmula, el nostre error és **e** = 0 – 1 = **-1**. En el cas que l’error fos 0, significaria que el perceptró hauria donat la resposta desitjada, per tant no caldria entrenar-lo; però com que en aquest cas no és 0, sí que cal entrenar-lo. La fórmula per variar el pes (o *weight*)és:

**wi = wi + wi**

a on wi és l’increment del pes, que pot ser tant positiu com negatiu. Per tant ara queda calcular aquest increment. La fórmula de l’increment és:

**wi = error \* *input* \* *learning rate***

a on error és l’error del perceptró i *input* és la senyal d’entrada d’aquell específic pes. També hi ha un terme nou anomenat *learning rate*, que és un nombre entre 0 i 1, i indica en quina proporció ha de variar el pes. Aquest terme serveix poder canviar la velocitat en la que arribes al pes adequat, tenint en compte que quan més ràpid, més imprecís i viceversa.

És important saber que per l’aprenentatge supervisat cal tenir tres grups grans de dades:

1. Dades d’entrenament: Són una sèrie de senyals d’entrada amb les seves respectives respostes. Aquestes dades són exclusives per entrenar el perceptró i corregir-lo si fes falta.
2. Dades de prova: Són una sèrie de senyals d’entrada de les quals també sabem la resposta, però no s’usen per entrenar el perceptró, sinó per veure si l’entrenament està sent efectiu o no.
3. Dades desconegudes: Són una sèrie de dades de les quals no saps la resposta que s’introduiran al perceptró un cop ja estigui entrenat per saber la resposta

És molt important tenir aquests tres grups de dades, sobretot el primer i el segon, ja que si només tinguéssim dades per entrenar el perceptró i no en tinguéssim per comprovar si l’entrenament va bé, podria passar que sobre entrenéssim el perceptró i només funcionés amb les dades d’entrenament i que no tornés una resposta correcta amb les dades desconegudes.

**Usos del perceptró**

Ara que ja hem vist com funciona un perceptró, des de que introduïm dades fins que l’entrenem i el millorem, cal veure les seves principals aplicacions, les quals són prediccions i classificacions.

Però abans, hem d’introduir un nou element: el *bias.* Si mirem a la fórmula **z = x \* w**, podem arribar a la conclusió que si x = 0, la resposta sempre serà 0, no importa el valor del pes. Però hi haurà casos en els que amb x = 0 voldrem una resposta que no sigui 0, per això serveix el *bias*. Si implementem el *bias*, la fórmula queda **z = x \* w + b**

1. **Perceptró per prediccions**

Imaginem que tenim un plànol cartesià a on a l’eix de les x tenim els metres quadrats d’una casa i a l’eix de les y tenim el seu preu. Si recol·lectem gran quantitat de dades, podrem veure que cada un dels punts sembla que estigui sobre una recta. Si entrenéssim un perceptró amb totes aquelles dades, ell mateix trobaria la fórmula de la recta, i per tant, podria fer prediccions.



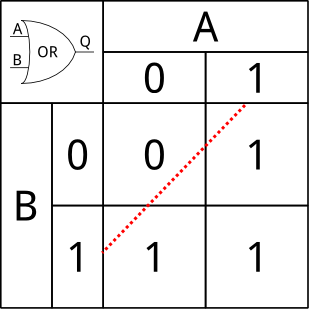
El perceptró pot fer això gràcies a la similitud entre la seva fórmula z = x·w +b i la fórmula de la recta y = mx + n. Al entrenar el perceptró amb aquelles dades, el que fem és buscar el pes i el *bias*, que corresponen al pendent i al punt d’intersecció de la recta amb l’eix de les ordenades respectivament. Per tant, quan ja està ben entrenat, podem introduir-li metres quadrats d’una casa i que ens torni una predicció del seu preu.

1. **Perceptró per classificacions**

Una altra aplicació comú pel perceptró és crear una divisió per poder classificar. De la mateixa manera que en les prediccions, el perceptró crea una funció de recta, i quan li introduïm noves dades ens pot dir si pertany al grup de sobre la recta o al grup de sota.



Aquest tipus de perceptró és el que s’usa, per exemple, en l’exemple anterior de les portes lògiques, ja que si mirem a la taula de veritat, veurem que podem traçar una línia per dividir els resultats que són zero i els resultats que són 1.

**Limitacions del perceptró**

Una de les grans limitacions del perceptró és que, com hem vist en els exemples anteriors, només funciona amb dades que són “linearment separables”. Si tenim una taula de veritats d’una porta lògica en la qual necessitem dues línies per separar el resultat o un model de prediccions que les dades no es poden predir amb una recta, un simple perceptró no podria fer cap de les dues tasques per molt que l’entrenéssim.

És per això que es va inventar el *multylayer perceptron*  o el perceptró de múltiples capes; també anomenat xarxa neuronal.