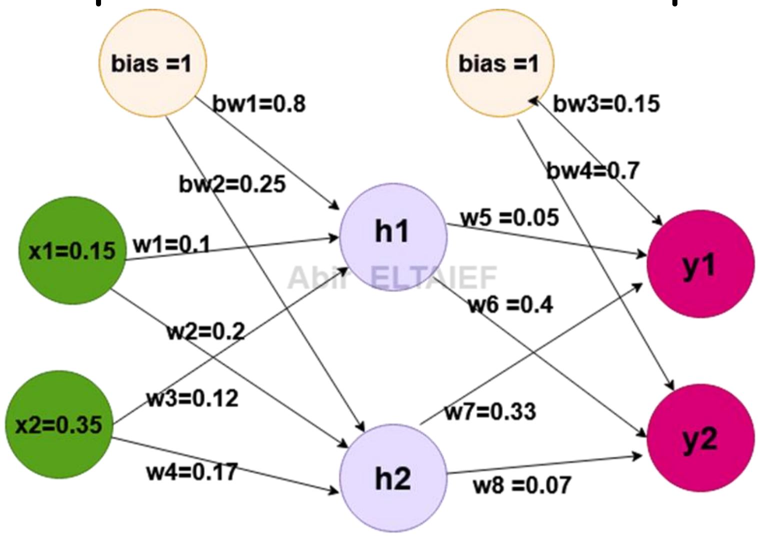
1- L'exemple d'un réseau de neurones simple :



Imaginons qu'il s'agit de prédire s'il s'agit d'une pomme donc output [1,0], ou tomate [0,1]

2- Propagation et calculs :

****** Aller de « l'input layer » à la « hidden layer »**

En appliquant le principe de la propagation illustré dans la 1ère figure, ci dessous, on trouve le premier résultat (2ème figure), puis en appliquant la fonction d'activation, on trouve l'output final (3ème figure) :

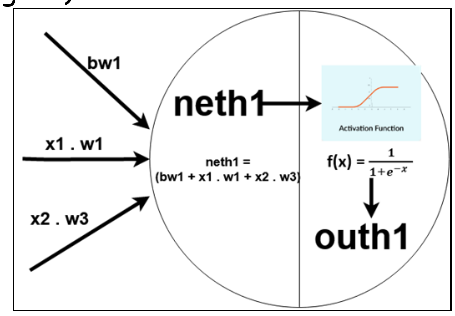


Figure1

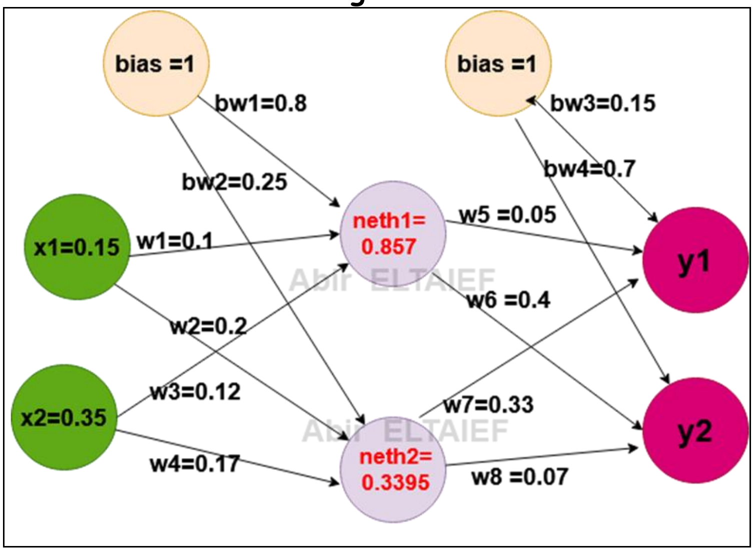


Figure2

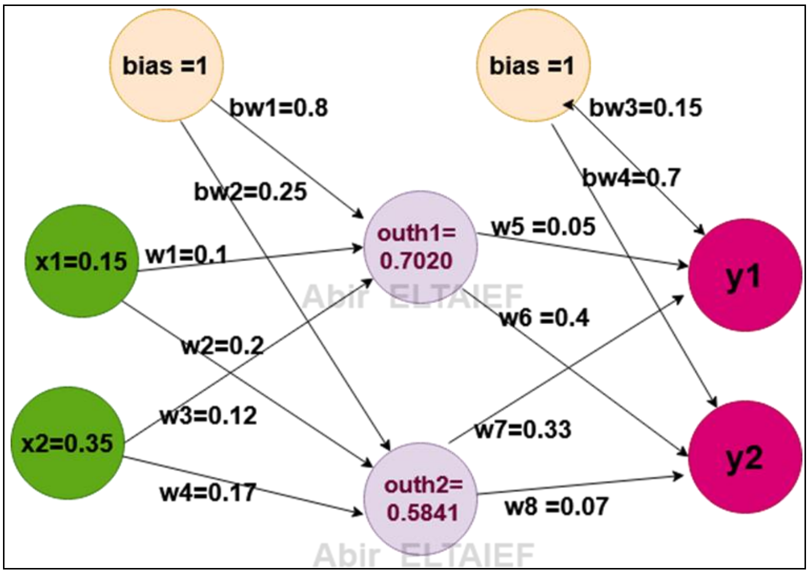
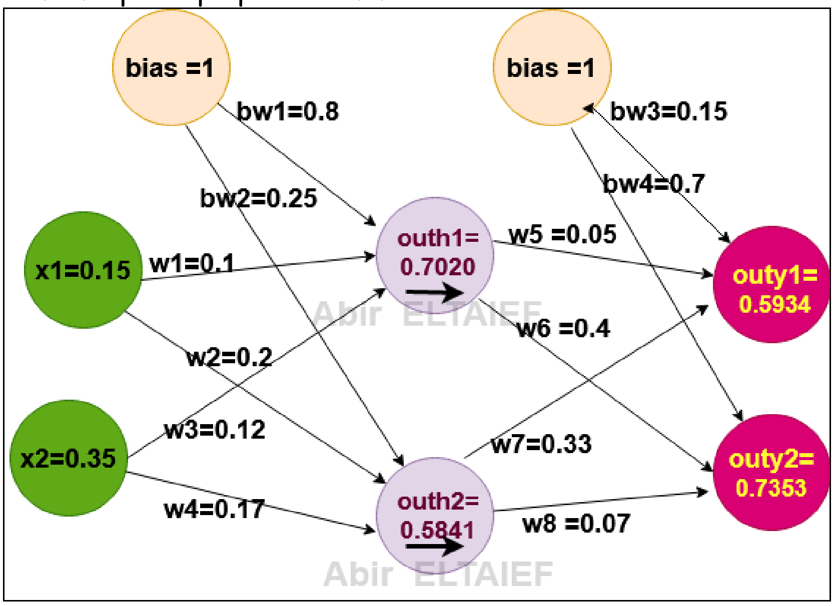


Figure3

****** Aller de la « hidden layer » à la « output layer » :**
En appliquant le même principe précédemment : on trouve donc :



3- Calcul de l'erreur totale

Etotale= ½ ((y1 - out y1)² + (y2 - outy2)²)

$$E_{totale} = \frac{1}{2}\left((y_1 - out\,y_1)^2 + (y_2 - outy_2)^2\right)$$

Dans notre cas, nous supposons que notre instance est une totmate, donc [0,1], ainsi :

$$E_{totale} = 0.5 \times ((0-0.5934)^2 + (1-0.7353)^2) = \mathbf{0.21105}$$

4- Rétropropagation (Calcul des gradients):

Etape1 : calculer les gradients pour les poids extérieurs (entre hidden layer et ouput layer) : w5,w6,w7 & w8):

*********on commence par w5:*****

******* premier « unpacking »:***

∂Etotal / ∂w5 = (**∂Etotal / ∂outy1) x (∂outy1/ ∂w5)**

- *avec* ***outy1*** *est l'output du nœud y1 (après l'application de la fonction d'activation)*

******* 2ème « unpacking »:***

∂Etotal / ∂w5 =(**∂Etotal / ∂outy1) x(∂outy1/ ∂nety1) x(∂nety1/ ∂w5)**

- *avec* ***nety1*** *est l'input du nœud y1 avant d'appliquer la fonction d'activation.*

Etotal = Ey1 +Ey2
Ey1 = ½ (targety1 - outy1)²
=> Etotal = ½ (targety1 - outy1)² + ½ (targety2 - outy2)²

******* Calculer les trois dérivées partielles:***

- **1ère dérivée partielle:**

∂Etotal / ∂outy1 = 2 x (½) x (targety1 - outy1) x(-1) + 0
∂Etotal / ∂outy1 = (outy1 - targety1) = 0.5934 - 0 = 0.5934

- **2ème dérivée partielle:**

∂outy1/ ∂nety1= ?

Il faut examiner la fonction d'activation qui transforme nety1 en outy1 : f
outy1= 1/(1+ exp(-nety1)) = f(nety1)

Sans trop entrer dans les détails (vous pouvez le démontrer ou voir dans des références mathématiques…)

∂(f(nety1))/ ∂nety1= f(nety1) . (1 - f(nety1))
= outy1 (1-outy1) = 0.5934 *(1 - 0.5934) = 0.2413

- **3ème dérivée partielle :**

La plus simple :
∂nety1/ ∂w5 = outh1 = 0.7020

=> ∂Etotal / ∂w5 = (outy1 - targety1) x outy1 (1-outy1) x outh1
= 0.5934 x 0.2413 x 0.7020 = 0.1005

*******On continue le même calcul avec w6:***

Dans ce cas , y2 dépend uniquement de w6 (pas y1 qui en dépend)

∂Etotal / ∂w6 =(∂Etotal / ∂outy2) x(∂outy2/ ∂nety2) x(∂nety2/ ∂w6)
= (outy2 - targety2) x outy2 (1-outy2) x outh1
= (0.7353 - 1) x 0.7353 x (1- 0.7353) x 0.7020 = - 0.0362

******* On continue le même calcul avec w7:***

Dans ce cas , y1 dépend uniquement de w6 (pas y2 qui en dépend)

∂Etotal / ∂w7 =(∂Etotal / ∂outy1) x(∂outy1/ ∂nety1) x(∂nety1/ ∂w7)
= (outy1 - targety1) x outy1 (1-outy1) x outh2
= (0.5934-0) x 0.5934(1-0.5934) x 0.5841 = 0.0836

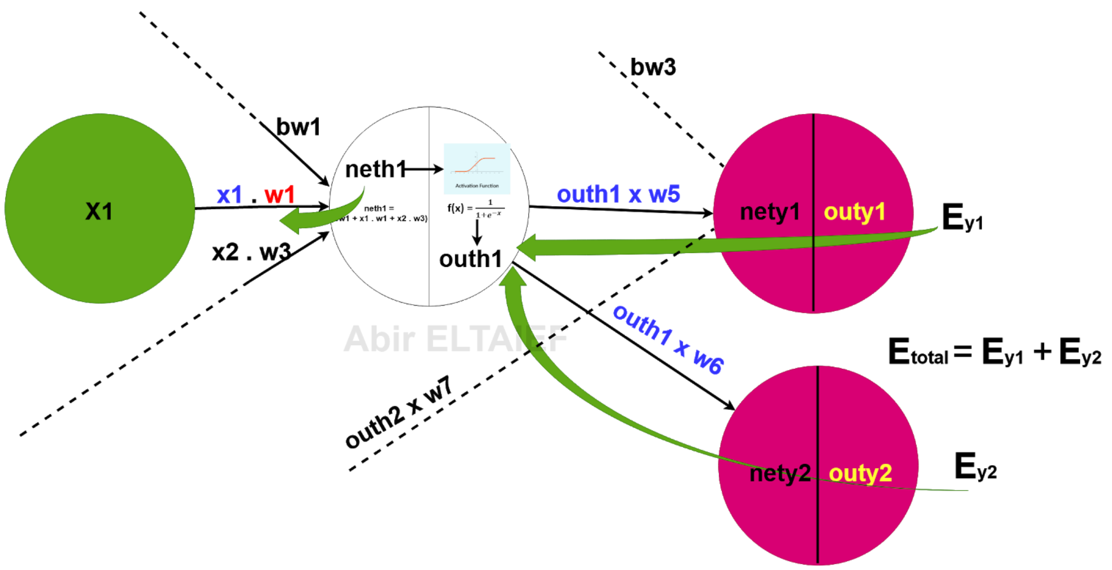
******* On continue le même calcul avec w8:***

Dans ce cas , y2 dépend uniquement de w6 (pas y1 qui en dépend)
∂Etotal / ∂w8 =(∂Etotal / ∂outy2) x(∂outy2/ ∂nety2) x(∂nety2/ ∂w8)
= (outy2 - targety2) x outy2 (1-outy2) x outh2
= (0.7353 - 1) x 0.7353(1-0.7353) x 0.5841 = -0.0301

Etape2: calculer les gradients pour les poids intérieurs (entre input layer et hidden layer) : (w1 ,w2,w3 & w4):

*******on commence par w1 :***

Ici, pour les poids intérieurs : le calcul est différent car chaque neurone du hidden layer, contribue aux deux ouputs ! ci après une figure :



∂Etotal / ∂w1 =(∂Etotal / ∂outh1)x(∂outh1/ ∂neth1) x(∂neth1/ ∂w1)

__* 1er unpacking (1ère dérivée partielle):__***

∂Etotal / ∂outh1 = (∂Ey1 / ∂outh1) + (∂Ey2 / ∂outh1)
***1er terme:
∂Ey1 / ∂outh1 = (∂Ey1 / ∂outy1) x (∂outy1 / ∂outh1)
= (∂Ey1 /∂outy1) x (∂outy1 /∂nety1) x (∂nety1 /∂outh1)
= (outy1 - targety1) x outy1(1-outy1) x w5
= (0.5934 - 0) x 0.5934 x(1- 0.5934) x 0.05 = 0.1432 x 0.05 = 0.0072

***2ème terme:
∂Ey2 / ∂outh1 = (∂Ey2 / ∂outy2) x (∂outy2 / ∂outh1)
= (∂Ey2 /∂outy2) x (∂outy2 /∂nety2) x (∂nety2 /∂outh1)
= (outy2 - targety2) x outy2(1-outy2) x w6
= (0.7353 - 1) x 0.7353 (1-0.7353) x 0.4
= -0.0206
__* 2ème unpacking (2ème dérivée partielle):__***
∂outh1/ ∂neth1 = outh1 x (1-outh1) = 0.7020 x (1 - 0.7020) = 0.2092

__* 3ème unpacking (3ème dérivée partielle):__***
∂neth1/ ∂w1= x1 = 0.15

=> ∂Etotal / ∂w1= (0.0072 + (-0.0206)) x 0.2092 x 0.15 =
= -0.0134 x 0.2092 x 0.15 = - 0.000420492

Selon les calculs précdents :
∂Etotal / ∂w1 =
[(outy1 - targety1)x outy1(1-outy1) x w5 +(outy2 - targety2) x outy2(1-outy2) x w6] x outh1 x (1-outh1) x x1

__*On continue même traitement avec w2:__***
Selon la formule ci-dessus :

∂Etotal / ∂w2 =(∂Etotal / ∂outh2)x(∂outh2/ ∂neth2) x(∂neth2/ ∂w2)

=[(outy1- targety1) x outy1(1-outy1) x w7 + (outy2- targety2) x outy2(1-outy2) x w8] x outh2 x (1 -outh2) x x1
= [(0.5934-0) x0.5934 x (1- 0.5934) x 0.33 + (0.7353-1) x 0.7353 (1- 0.7353) x 0.07] x 0.5841 x (1- 0.5841) x 0.15
= 0.0437 x 0.2429 x 0.15
= 0.00159221

__* On continue même traitement avec w3:__***

∂Etotal / ∂w3 =(∂Etotal / ∂outh1)x(∂outh1/ ∂neth1) x(∂neth1/ ∂w3)
=[(outy1- targety1) x outy1(1-outy1) x w5 + (outy2- targety2) x outy2(1-outy2) x w6] x outh1 x (1 -outh1) x x2
= -0.0134 x 0.2092 x 0.35
= - 0.000981148

******* On continue même traitement avec w4:***

∂Etotal / ∂w4 =(∂Etotal / ∂outh2)x(∂outh2/ ∂neth2) x(∂neth2/ ∂w4)
= [(outy1- targety1) x outy1(1-outy1) x w7 + (outy2- targety2) x outy2(1-outy2) x w8] x outh2 x (1 -outh2) x x2

= 0.0437 x 0.2429 x 0.35
= 0.003715156

Etape3: calculer les gradients des « bias weights » de l'extérieur (bw3 & bw4):

*******on comence par bw3:***
∂Etotal / ∂bw3 = (∂Etotal / ∂outy1) x (∂outy1/ ∂nety1) x (∂nety1/ ∂bw3)
= (outy1 - targety1) x outy1 (1- outy1) x 1
= 0.1432
*******on continue avec bw4:***
∂Etotal / ∂bw4 = (∂Etotal / ∂outy2) x (∂outy2/ ∂nety2) x (∂nety2/ ∂bw4)
= (outy2 - targety2) x outy2 (1- outy2) x 1
= -0.0515

Etape4: calculer les gradients des « bias weights » de l'intérieur (bw1 & bw2):

*******on commence par bw1 :***
∂Etotal / ∂bw1 = (∂Etotal / ∂outh1) x (∂outh1 / ∂neth1) x (∂neth1 / ∂bw1)
=[(outy1- targety1) x outy1(1-outy1) x w5 + (outy2- targety2) x outy2(1-outy2) x w6] x outh1 x (1 -outh1) x 1
= -0.0134 x 0.2092
= - 0.0028

*******on continue le même traitement avec bw2:***
∂Etotal / ∂bw2 = (∂Etotal / ∂outh2) x (∂outh2 / ∂neth2) x (∂neth2 / ∂bw2)
= [(outy1- targety1) x outy1(1-outy1) x w7 + (outy2- targety2) x outy2(1-outy2) x w8] x outh2 x (1 -outh2) x 1

= 0.0437 x 0.2429
= 0.0106
Etape 5: Résumé

∂Etotal / ∂w1= - 0.000420492	∂Etotal / ∂w2= 0.00159221	∂Etotal / ∂w3= -0.000981148	∂Etotal / ∂w4= 0.003715156
∂Etotal / ∂w5= 0.1005	∂Etotal / ∂w6= - 0.0362	∂Etotal / ∂w7= 0.0836	∂Etotal / ∂w8= -0.0301
∂Etotal / ∂bw1= - 0.0028	∂Etotal / ∂bw2= 0.0106	∂Etotal / ∂bw3= 0.1432	∂Etotal / ∂bw4= -0.0515

5- Mise à jour des poids

<u>=> Results for first iteration:</u>

- w1_new = w1 - 0.5 x (∂Etotal / ∂w1) = 0.1 - 0.5 x(- 0.000420492) = 0.1002
- w2_new = w2 - 0.5 x (∂Etotal / ∂w2) = 0.2 - 0.5 x (0.00159221) = 0.1992
- w3_new = w3 - 0.5 x (∂Etotal / ∂w3) = 0.12 - 0.5 x (-0.000981148) = 0.1205
- w4_new = w4 - 0.5 x (∂Etotal / ∂w4) = 0.17 - 0.5 x (0.003715156) = 0.1681
- w5_new = w5 - 0.5 x (∂Etotal / ∂w5) = 0.05 - 0.5 x (0.1005) = 0.00025
- w6_new = w6 - 0.5 x (∂Etotal / ∂w6) = 0.4 - 0.5 x (- 0.0362) = 0.4181
- w7_new = w7 - 0.5 x (∂Etotal / ∂w7) = 0.33 - 0.5 x (0.0836) = 0.2882
- w8_new = w8 - 0.5 x (∂Etotal / ∂w8) = 0.07 - 0.5 x (-0.0301) = 0.0851

- bw1_new = bw1 - 0.5 x (∂Etotal / ∂bw1) = 0.8 - 0.5x (- 0.0028) = 0.8014
- bw2_new = bw2 - 0.5 x (∂Etotal / ∂bw2) = 0.25 - 0.5x (0.0106) = 0.2447
- bw3_new = bw3 - 0.5 x (∂Etotal / ∂bw3) = 0.15 - 0.5 x (0.1432) = 0.0784
- bw4_new = bw4 - 0.5 x (∂Etotal / ∂bw4) = 0.7 - 0.5 x (-0.0515) = 0.7258

⇨ Dans le monde réel il faudrait répéter le calcul des gradients et la mise à jour des poids jusqu'à ce que l'erreur totale atteint un seuil bien défini et que le réseau de neurones trouve la combinaison des poids qui permet d'atteindre ce seuil !

Abir ELTAIEF