RÉSEAUX NEURONAUX : UN APERÇU

PHI 6385

Séance 8

Jonathan Simon

PROGRAMME

- 1) Qu'est-ce qu'un réseau neuronal artificial? (ANN)
- 2) Le cas le plus simple : un perceptron
- 3) Les perceptrons en tant qu'implémentations de systèmes de symboles (portes NAND)
- 4) Activation sigmoïde et réseaux neuronaux capables d'apprendre
- 5) l'apprentissage des fonctions comme l'apprentissage de modèles prédictifs du monde

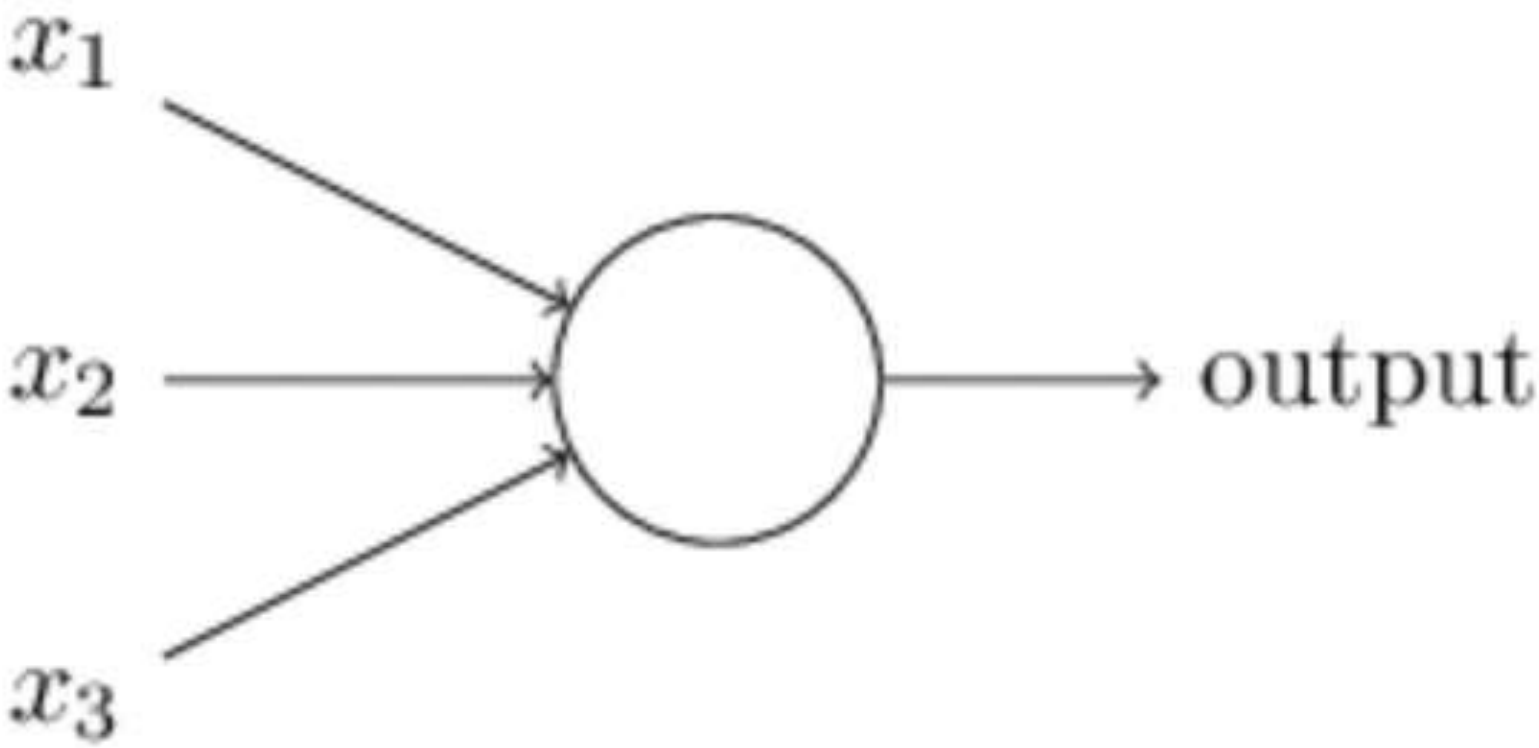
PROGRAMME

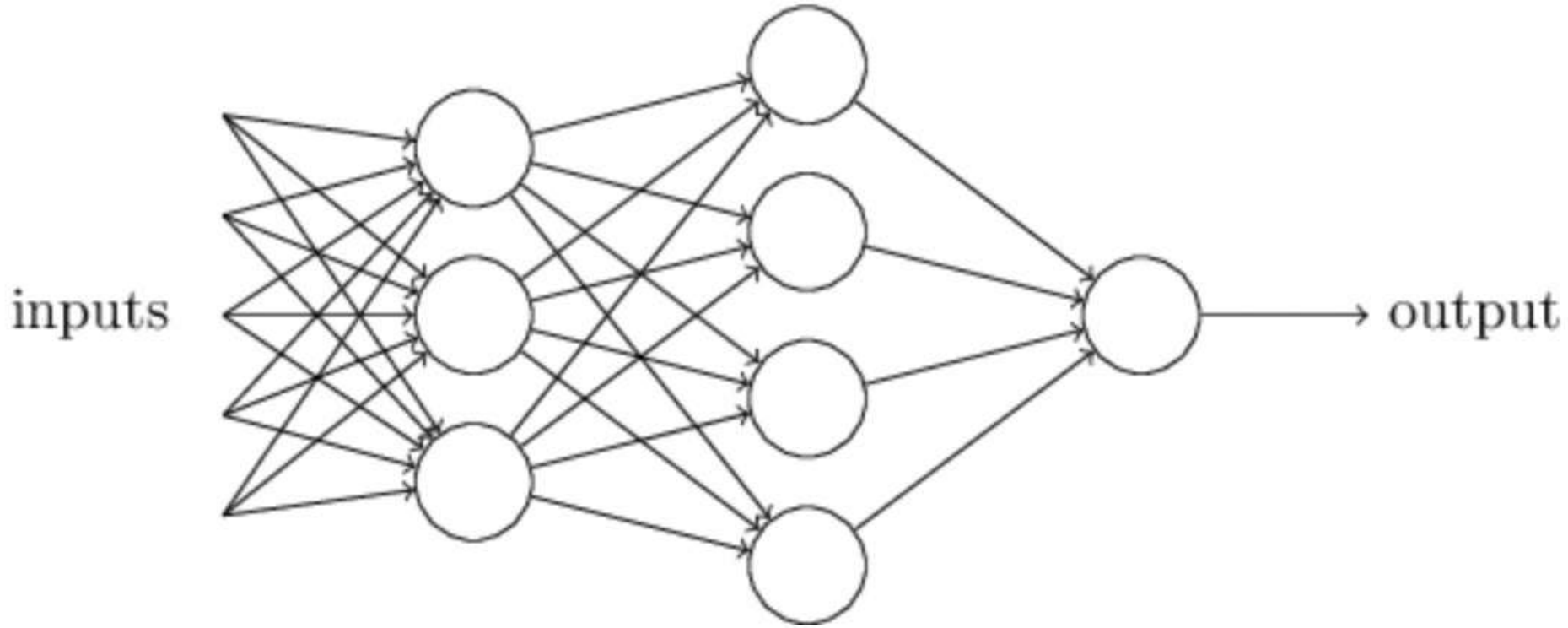
- 6) Le théorème d'approximation universelle : comment un réseau neuronal peut approximer toute fonction continue
- 7) Descente de gradient et backpropagation : comment les réseaux neuronaux apprennent
- 8) Pourquoi c'est difficile : pourquoi nous avons besoin de réseaux neuronaux profonds (réseaux neuronaux avec de nombreuses couches) et d'architectures compliquées pour permettre l'apprentissage de choses compliquées.

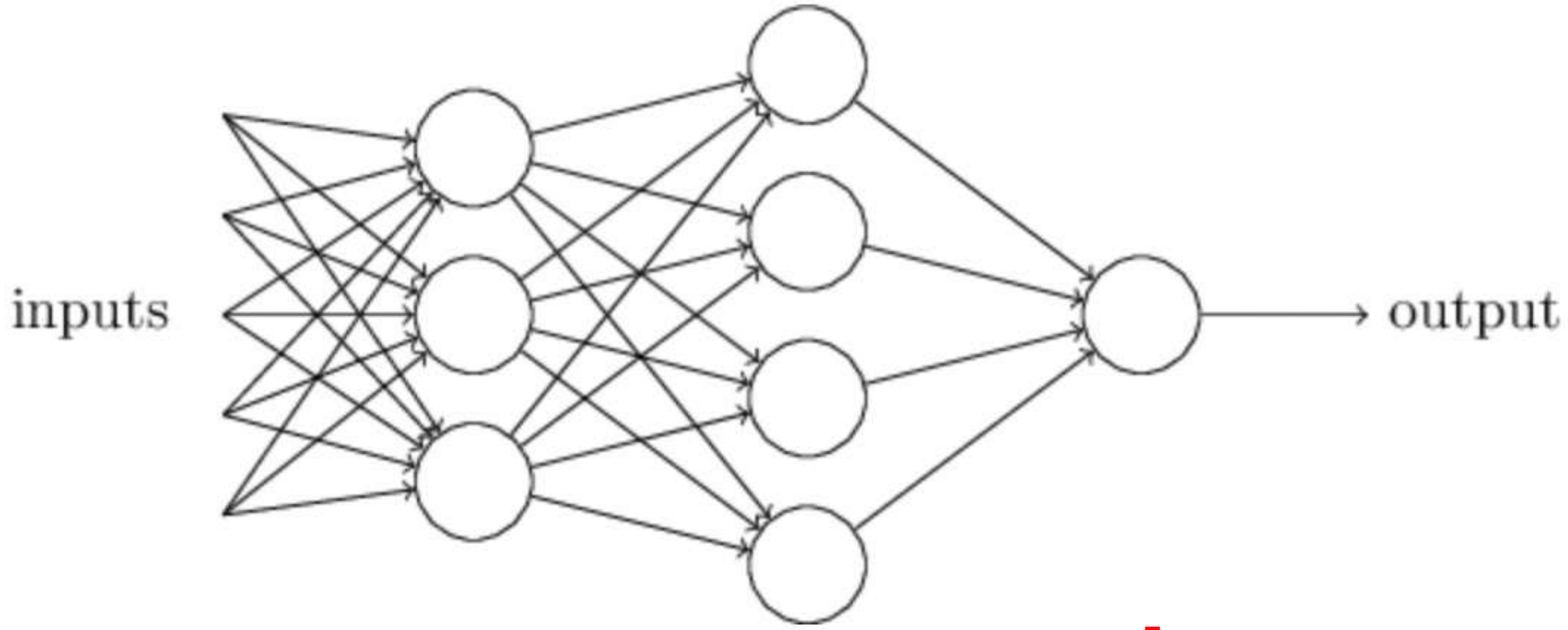
QU'EST-CE QU'UN RÉSEAU NEURONAL ARTIFICIAL? (ANN)

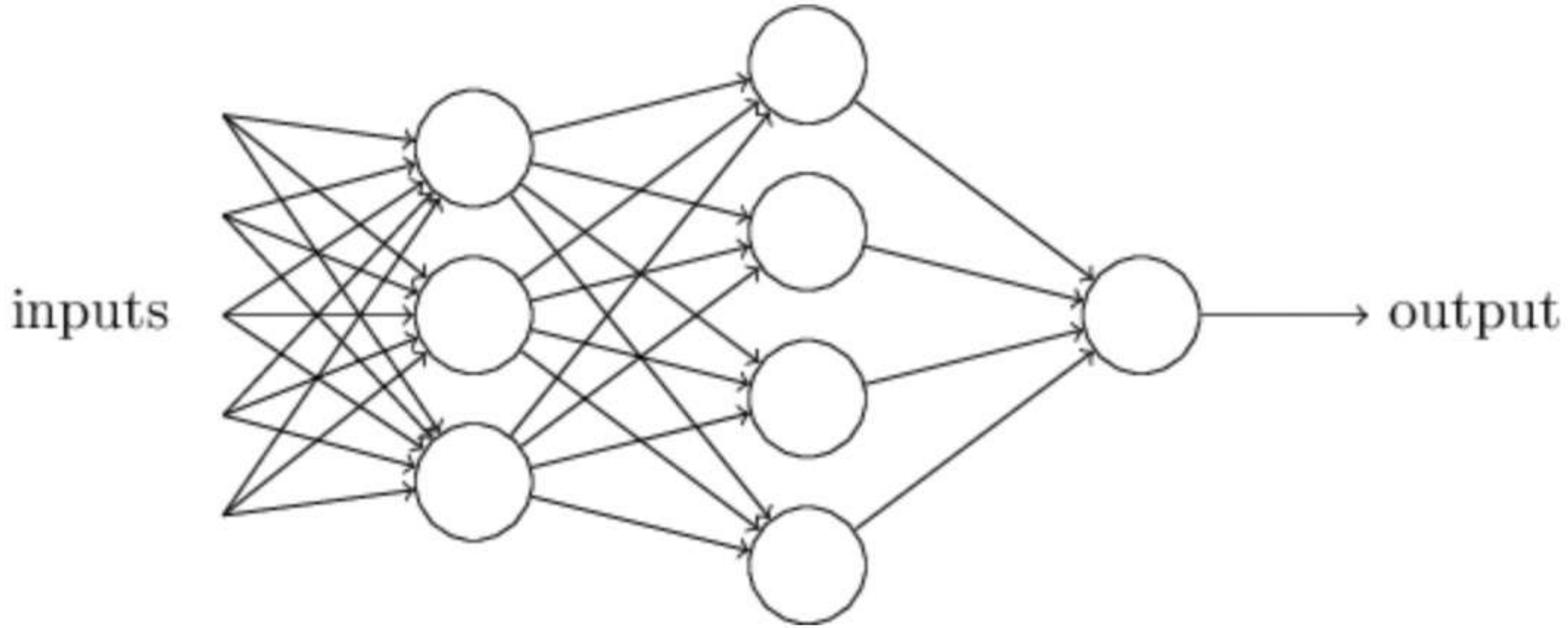
QU'EST-CE QU'UN RÉSEAU NEURONAL ARTIFICIAL? (ANN)

- Un réseau neuronal est une fonction complexe, composée de fonctions plus petites : F(G(x))
- Les plus petites fonctions sont les neurones : un neurone prend un nombre fixe d'entrées et délivre une sortie unique en fonction de ces entrées. (intuitivement : il décide de tirer ou non, en fonction de ses entrées)


x_1
x_2
x_3
output


inputs
output


inputs
output
X
H
G
F

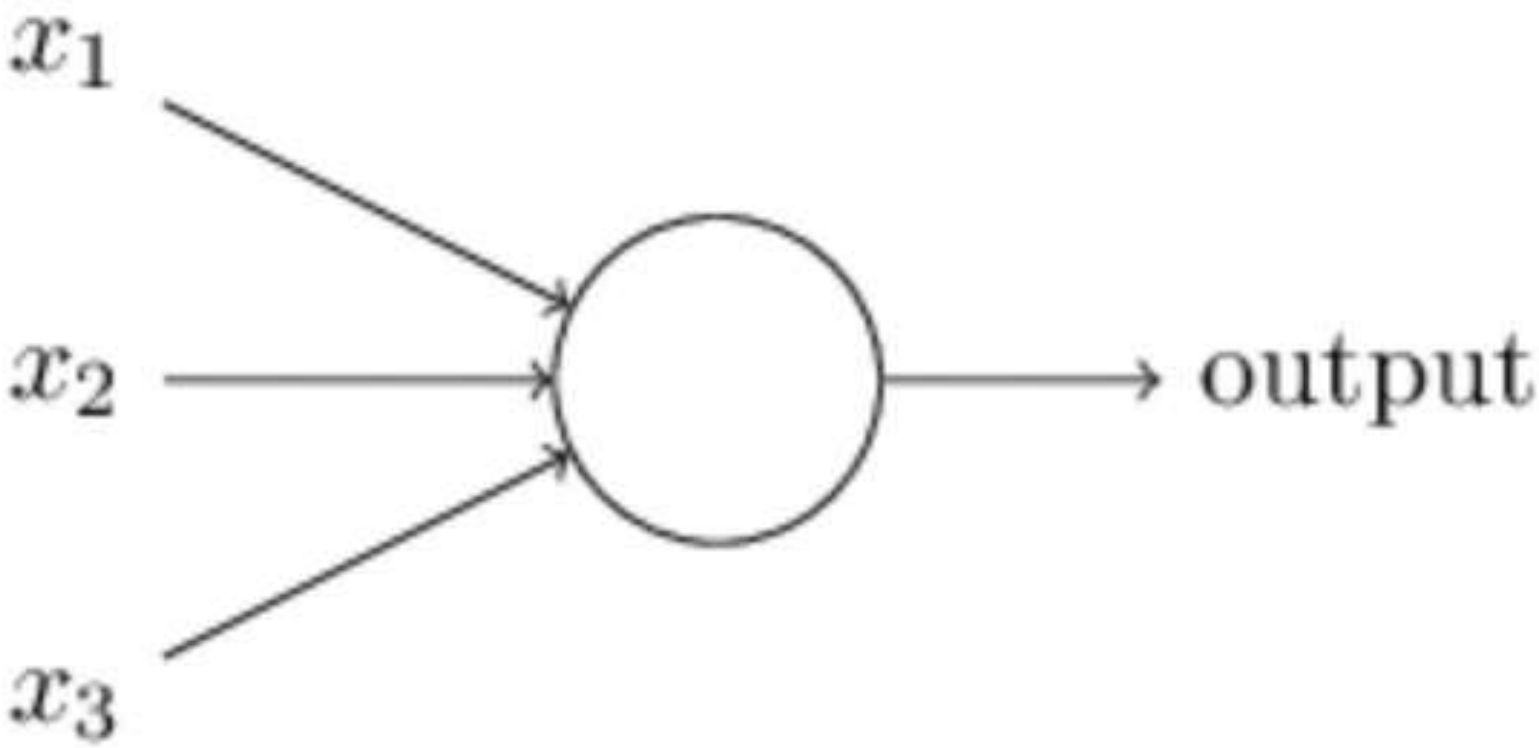


X H(X) G(H(X)) F(G(H(X)))

QU'EST-CE QU'UN RÉSEAU NEURONAL ARTIFICIAL? (ANN)

- Observez que ce sont des entités mathématiques : abstraites, plutôt qu'artificielles. En tant que telles, elles peuvent être mises en œuvre par des systèmes biologiques aussi bien que par des systèmes synthétiques.

LE CAS LE PLUS SIMPLE : UN PERCEPTRON

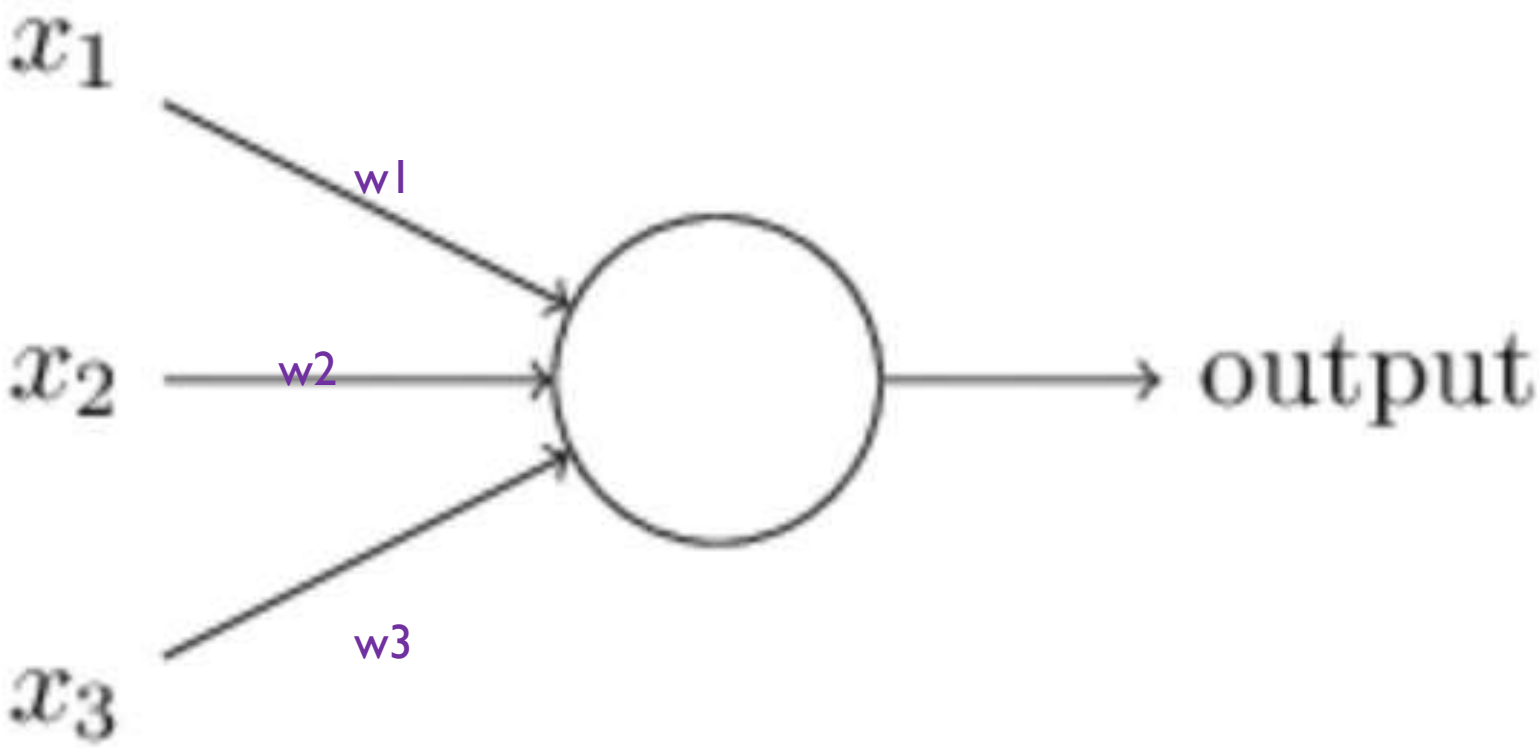

x_1
x_2
x_3
output

UN PERCEPTRON

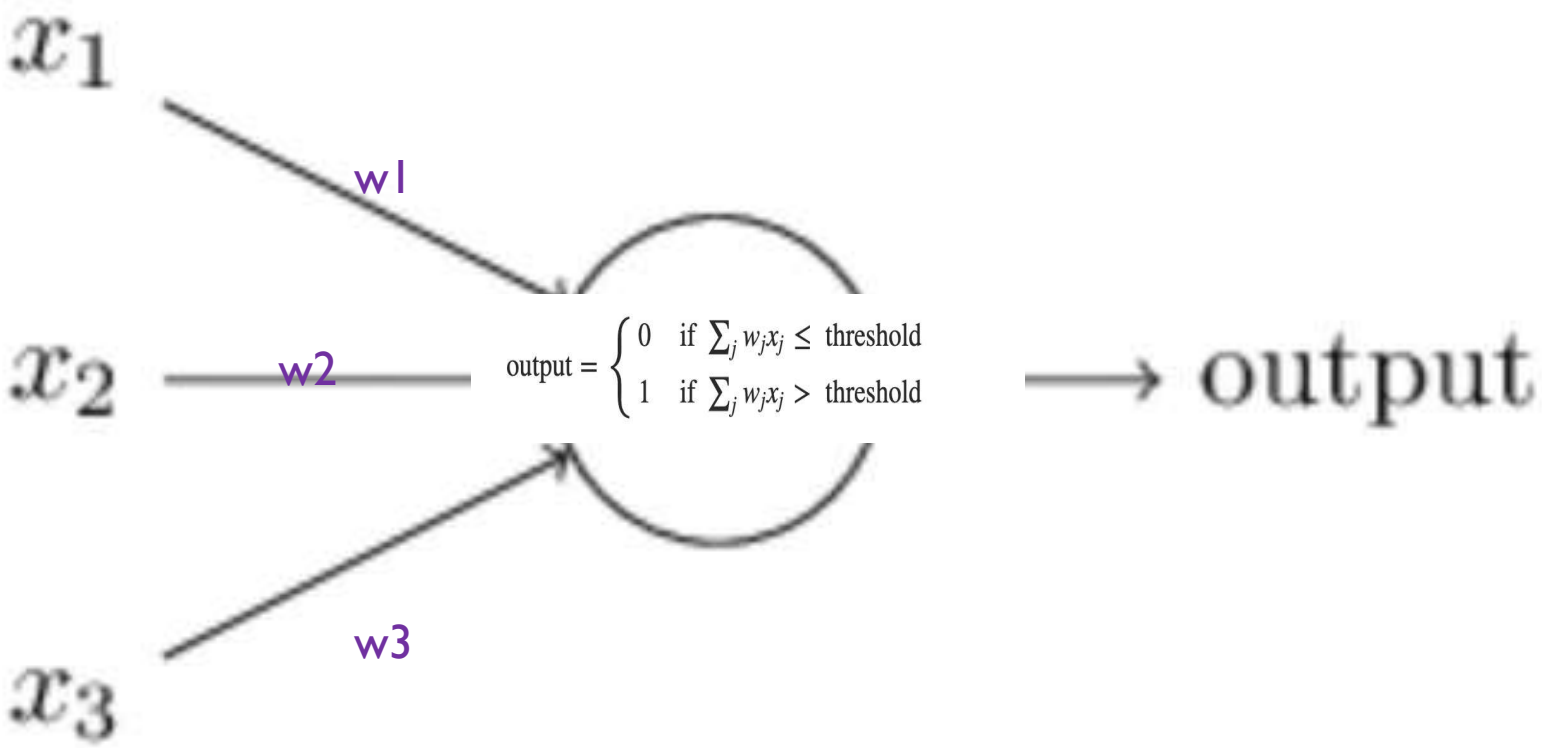
- Un perceptron est un neurone dont l'activation est fixée par un seuil : il sort 0 si le seuil n'est pas atteint, 1 s'il l'est.
- (note : cela signifie que les perceptrons ne sont pas différentiables, et que de très petits changements dans l'entrée conduisent à des sauts de 0 à 1 dans la sortie)

UN PERCEPTRON

- Nous comprenons également que le neurone attache un poids, *w*, à chaque entrée *x* : intuitivement, combien il se soucie de cette entrée


x_1
x_2
x_3
w1
w2
w3
output

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$


x_1
x_2
x_3
w1
w2
w3
output = $\begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$
output

UN PERCEPTRON

- Ainsi, pour décider s'il tire sur (x1,x2,x3), nous multiplions d'abord : x1 * w1, puis x.2 * w2, puis x3 * w3 ... puis nous les additionnons, et si la somme est supérieure à la valeur seuil, le neurone tire

- (j'utilize « * » pour symboliser multiplication)

UN PERCEPTRON

- Exemple : le neurone décide si vous allez au festival. Entrée 1 = x1 = probabilité qu'il va pleuvoir. Entrée 2 = probabilité que votre ami va venir. Entrée 3 = probabilité que ca va être facile de s'y rendre.

UN PERCEPTRON

- Peut-être que vous vous souciez surtout de la météo, alors w1 = 6. Cela ne vous dérange pas vraiment d'y aller seul, donc w2 = 2, et vous vous souciez moyennement du temps que cela prendra, donc w3 = 4.
- Alors si nous fixons le seuil à 5 :

UN PERCEPTRON

- Certain qu'il pleut, mais votre ami vient et c'est proche=

- (0 * 6) + (1 * 2) + (1 * 4) = 6, qui est plus grand que cinq, donc le neurone sort 1 = tu y vas

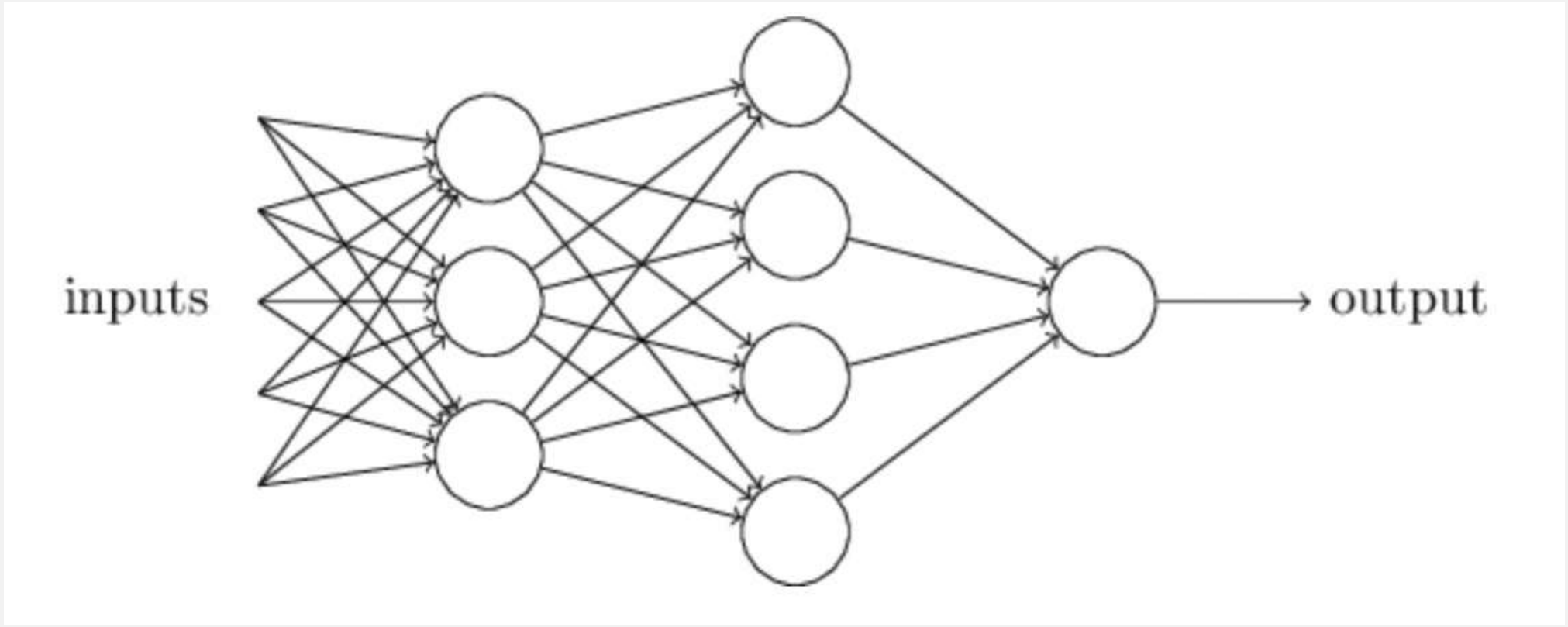
UN PERCEPTRON

- Plus typique : 85 % de chances de pluie, 80 % de chances que ton ami vienne, 75 % de chances qu'il n'y ait pas d'embouteillage=

- (.15 * 6) + (.8 * 2) + (.75 * 4) = 5.5, qui est plus grand que cinq, donc le neurone sort 1 = tu y vas

UN PERCEPTRON

- C'est l'idée générale. Notez que vous pouvez composer des perceptrons dans un réseau, ce qui permettra une hiérarchie de décisions plus complexes :


inputs
output

UN PERCEPTRON

- Peut-être que la deuxième couche de décision est de savoir si vous demandez plus d'argent à vos parents : un facteur est de savoir si vous allez au festival, un autre est de savoir si vous voulez acheter un nouvel ordinateur, etc... et à chacun de ces facteurs vous pourriez attribuer un poids de combien il devient alors plus important pour vous de demander de l'argent...

UN PERCEPTRON

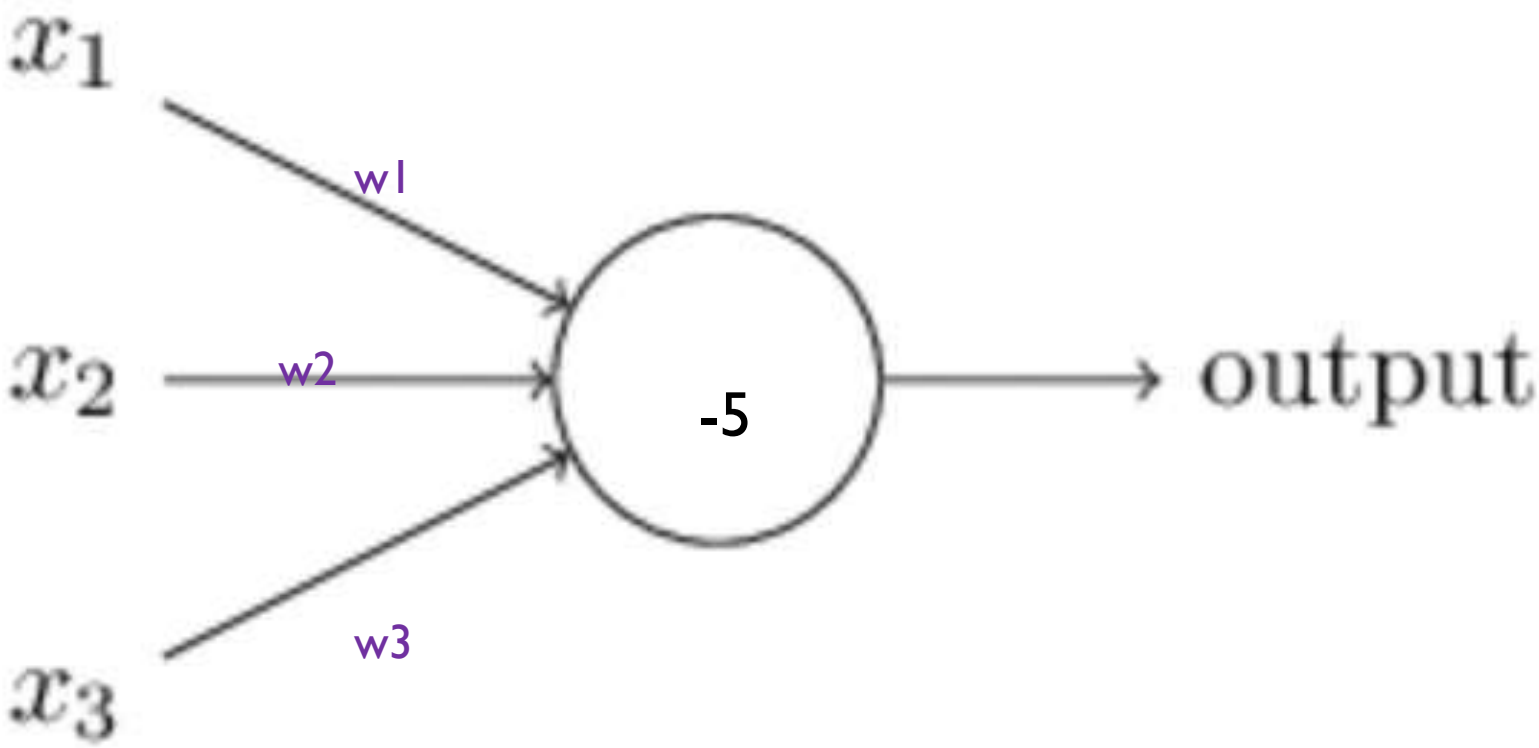
- Mais notez que si les entrées de ce deuxième niveau seront la sortie d'un premier niveau de perceptrons, alors ses entrées doivent toutes être 0 ou 1....

UN PERCEPTRON

- Au lieu de permettre au seuil d'être un nombre arbitraire, il peut être plus intuitif de penser que le seuil est zéro. Mais nous voulons toujours capturer l'idée que « la somme des termes doit être égale à 5 pour qu'il y ait déclenchement ». Si le seuil est zéro, cela signifie que nous avons besoin d'un terme supplémentaire de -5. C'est le *biais*.

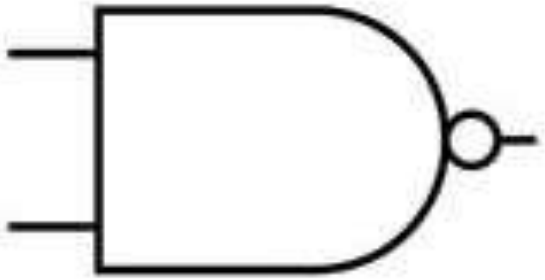
UN PERCEPTRON

- $(x1 * w1) + (x2 * w2) + (x3 * w3) \geq 5$
- Iff
- $(x1 * w1) + (x2 * w2) + (x3 * w3) \text{ -5} \geq 0$


x_1
w1
x_2
w2
-5
output
w3
x_3

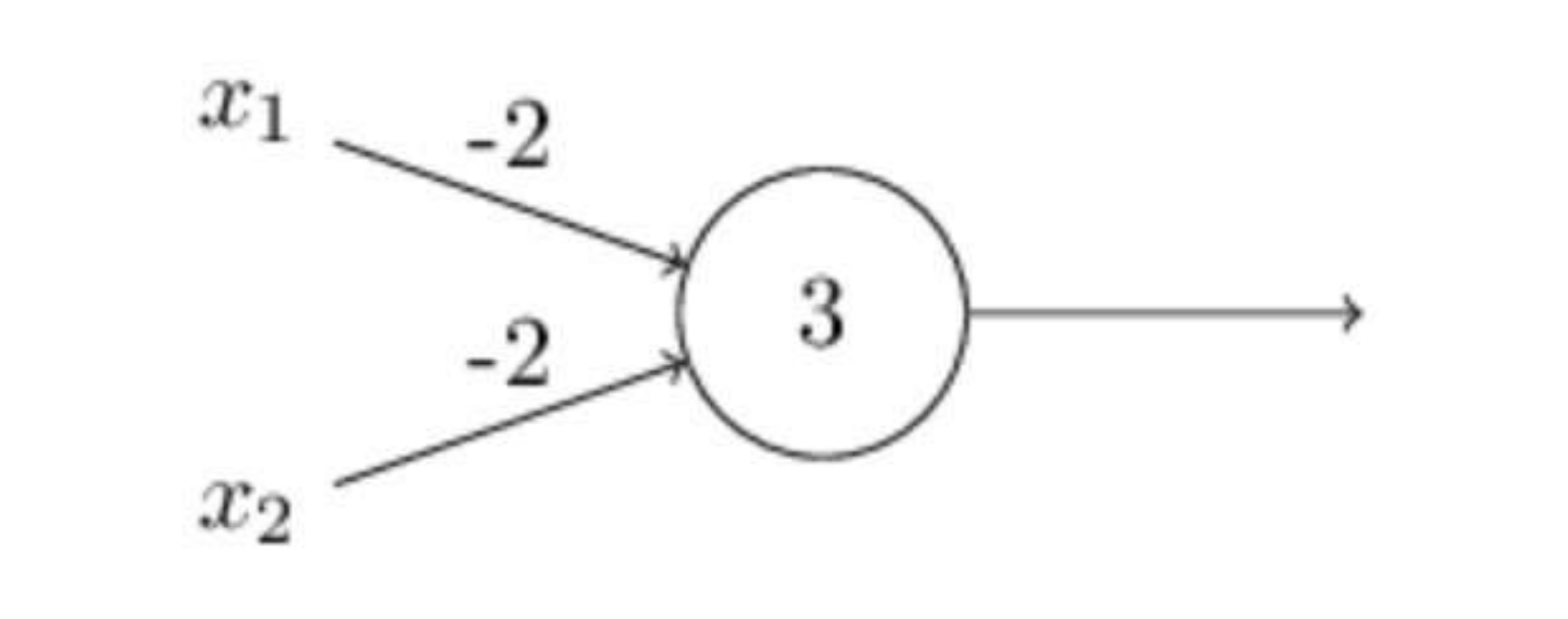
LES PERCEPTRONS EN TANT QU'IMPLÉMENTATIONS DE SYSTÈMES DE SYMBOLES (PORTES NAND)

NAND



A	B	Output
0	0	1
1	0	1
0	1	1
1	1	0

PERCEPTRONS EN TANT QU'IMPLEMENTATIONS



PERCEPTRONS EN TANT QU'IMPLEMENTATIONS

- Les poids de ce perceptron sont -2 et -2. Son biais est +3 .... en d'autres termes, son seuil est -3.

PERCEPTRONS EN TANT QU'IMPLEMENTATIONS

- Nous voyons alors que l'entrée 00 produit la sortie 1, puisque $(-2*0)+(-2*0)+3=3$ est positif.
- *Autrement dit :* $(-2*0)+(-2*0)=0$ est supérieur au seuil de -3, donc il se déclenche.
- Des calculs similaires montrent que les entrées 01 et 10 produisent la sortie 1.
- Mais l'entrée 11 produit la sortie 0, puisque $(-2*1)+(-2*1)+3=-1$ est négatif.
- Et donc notre perceptron implémente une porte NAND ! (NAND = tout sauf les deux)

PERCEPTRONS EN TANT QU'IMPLEMENTATIONS

- On peut montrer que toute porte logique peut être construite à l'aide de circuits NAND, donc les perceptrons peuvent implémenter des machines de Turing. (Cf. les affirmations de Fodor et Pylyshyn selon lesquelles ils ne sont peut-être bons qu'à cela)

PERCEPTRONS EN TANT QU'IMPLEMENTATIONS

- MAIS: Cependant, les réseaux neuronaux sont bons pour bien plus que cela (bien que les perceptrons ne le soient peut-être pas). Le secret réside dans la fonction d'activation.
- En passant d'une simple fonction d'activation à seuil à une fonction différentiable (c'est-à-dire qui ne saute pas de un à zéro), nous pouvons permettre l'apprentissage.

DES RÉSEAUX NEURONAUX CAPABLES D'APPRENDRE

DES RÉSEAUX NEURONAUX CAPABLES D'APPRENDRE

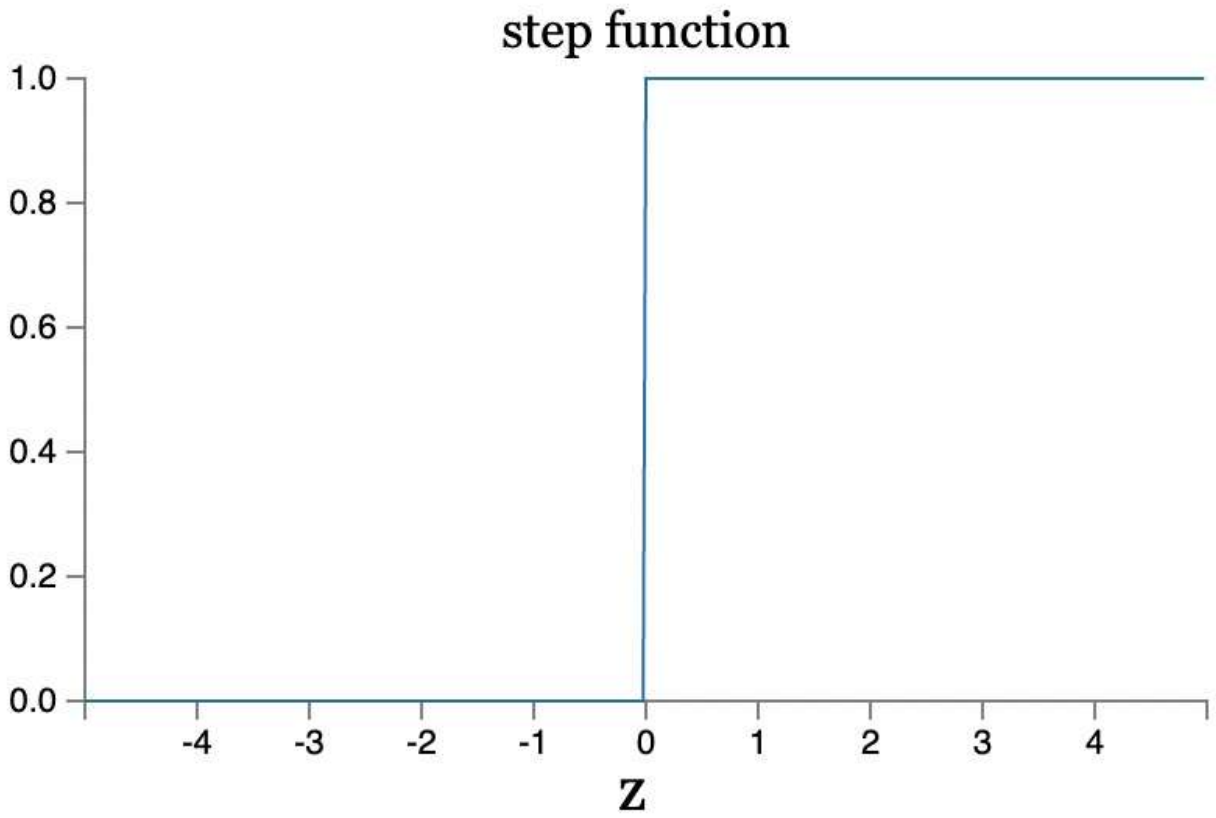
- Que serait l'apprentissage d'un réseau neuronal ? Disons que nous avons une certaine sortie cible, les sorties que nous voulons que le système fournisse.

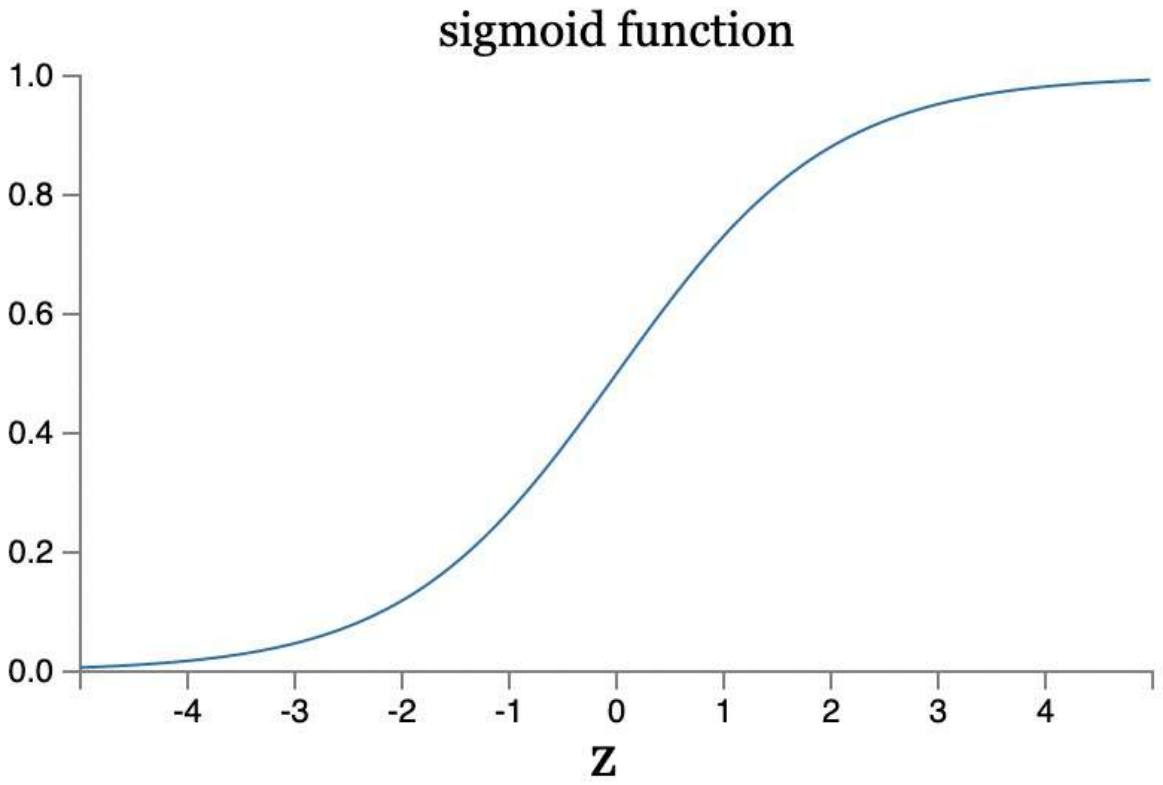
- L'apprentissage serait alors une méthode d'ajustement des poids et des biais du réseau (progressivement), jusqu'à ce que vous obteniez le résultat souhaité.

DES RÉSEAUX NEURONAUX CAPABLES D'APPRENDRE

- Le problème avec les perceptrons est que, comme ils ont des activations à seuil, de petites modifications de l'entrée peuvent produire de très grands changements dans la sortie.

- La solution : au lieu d'une véritable activation à seuil (fonction échelon), une fonction continue qui ne fait qu'approximer une fonction échelon (la fonction sigmoïde)


step function
1.0
0.8
0.6
0.4
0.2
0.0
-4
-3
-2
-1
0
1
2
3
4
z


sigmoid function
1.0
0.8
0.6
0.4
0.2
0.0
-4
-3
-2
-1
0
1
2
3
4
Z

$$\sigma(z) \equiv \frac{1}{1+e^{-z}}. \tag{3}$$

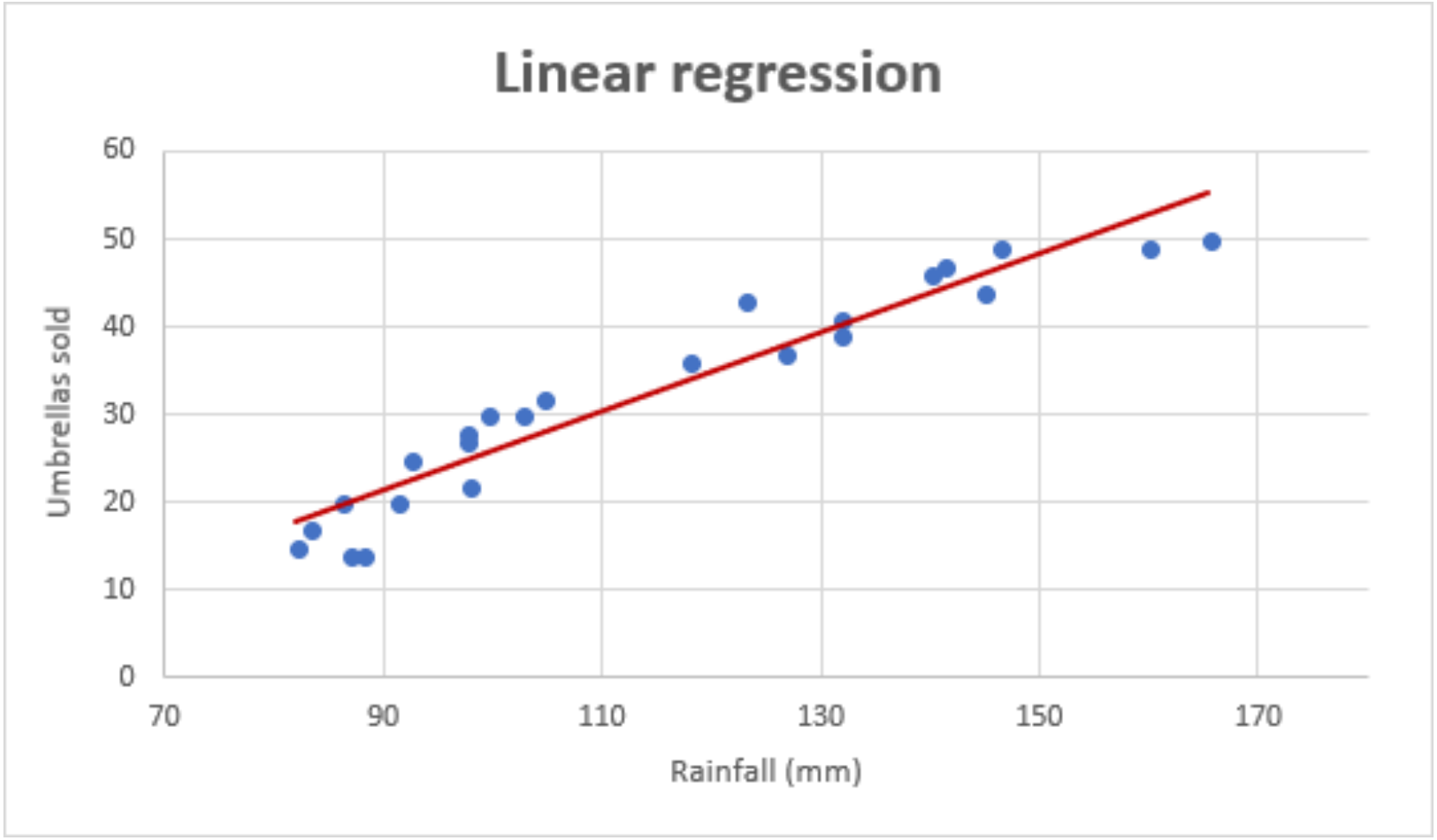
To put it all a little more explicitly, the output of a sigmoid neuron with inputs $x_1, x_2, \ldots$, weights $w_1, w_2, \ldots$, and bias b is

$$\frac{1}{1+\exp(-\sum_j w_j x_j - b)}. \tag{4}$$

DES RÉSEAUX NEURONAUX CAPABLES D'APPRENDRE

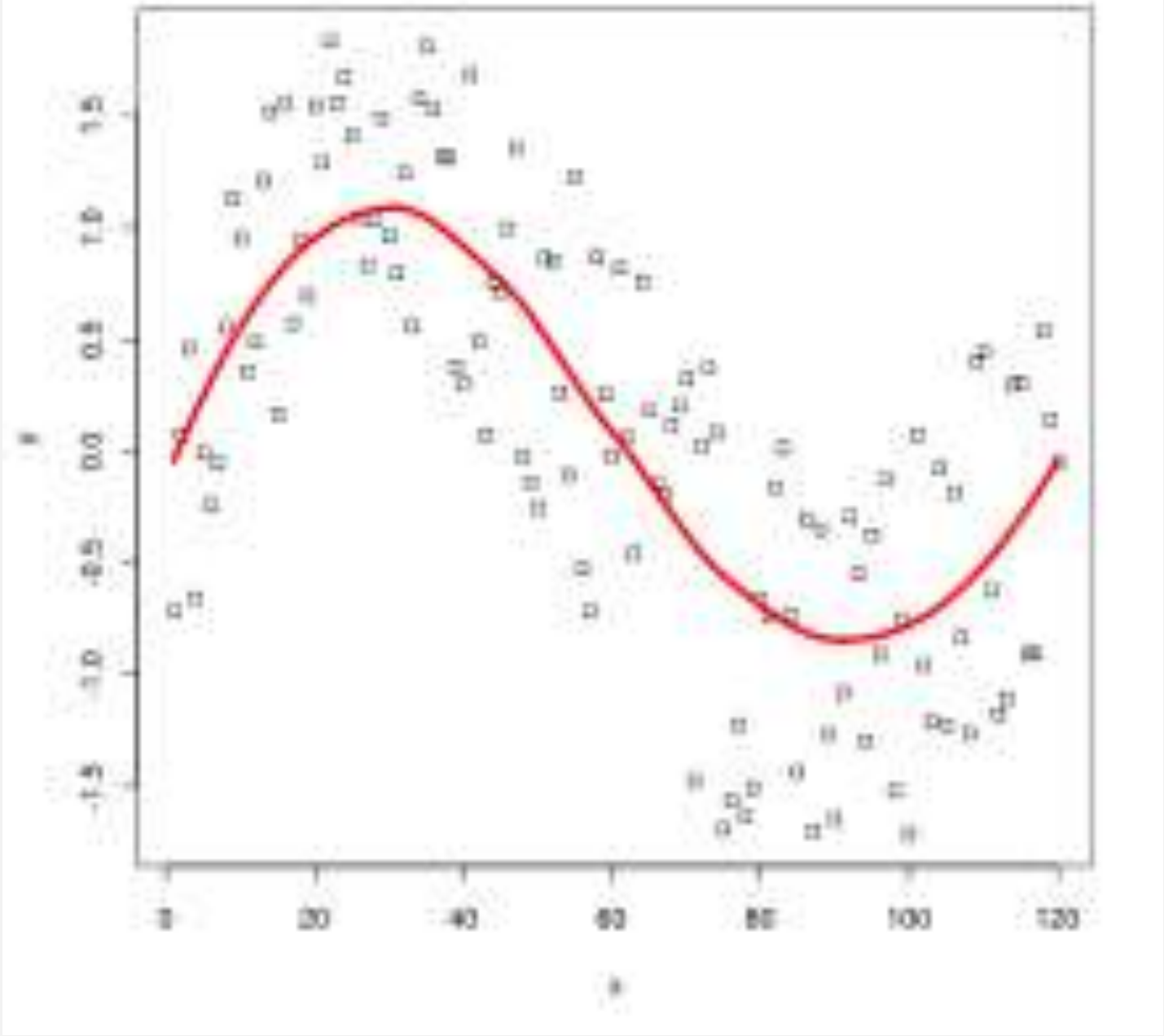
- En utilisant une telle fonction, nous pouvons nous assurer que des changements suffisamment petits dans les poids et les biais produiront des changements suffisamment petits dans la sortie, ce qui signifie que si nous pouvons trouver une méthode pour déterminer de quelles manières « pousser » les poids et les biais de notre réseau, nous pourrions lui faire apprendre une fonction de sortie souhaitée...

L'APPRENTISSAGE DES FONCTIONS COMME L'APPRENTISSAGE DE MODÈLES PRÉDICTIFS DU MONDE


Linear regression
Umbrellas sold
60
50
40
30
20
10
0
70
90
110
130
150
170
Rainfall (mm)

MODÈLES PRÉDICTIFS

- Erreur : la différence entre la prédiction et la valeur réelle
- Un bon modèle prédictif est une ligne ou une courbe (un hyperplan) qui minimise l'erreur (étant donné les degrés de liberté dont il dispose)


1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
0
20
40
60
80
100
120


Linear Regression Curve
Daily Chart - E-mini S&P 500 Future (ES)
Sell when Price deviates above
Linear Regression Curve
Sell
Buy
Sell
Buy
9-day Moving
Linear Regression
Buy when Price deviates below
Linear Regression Curve
Buy
Commodity.com - all rights reserved

MODÈLES PRÉDICTIFS

- Notez la possibilité d'un surajustement et d'un sous-ajustement : vous pouvez relier tous les points directement si vous avez suffisamment de paramètres ... mais cela ne fera pas une bonne projection en dehors de la distribution (surajustement / overfitting).

MODÈLES PRÉDICTIFS

- D'autre part, il est parfois évident que vous avez besoin d'une courbe ou d'une fonction périodique plutôt que d'une ligne droite, là, une ligne droite serait sous-adaptée. (c'est un art sombre que de savoir comment l'obtenir juste comme il faut cependant)

LE THÉORÈME D'APPROXIMATION UNIVERSELLE

LE THÉORÈME D'APPROXIMATION UNIVERSELLE

- Nous avons noté que les réseaux neuronaux pourraient être en mesure d'apprendre certaines fonctions, et qu'il existe certaines fonctions qu'ils pourraient apprendre. Il reste à établir :

LE THÉORÈME D'APPROXIMATION UNIVERSELLE

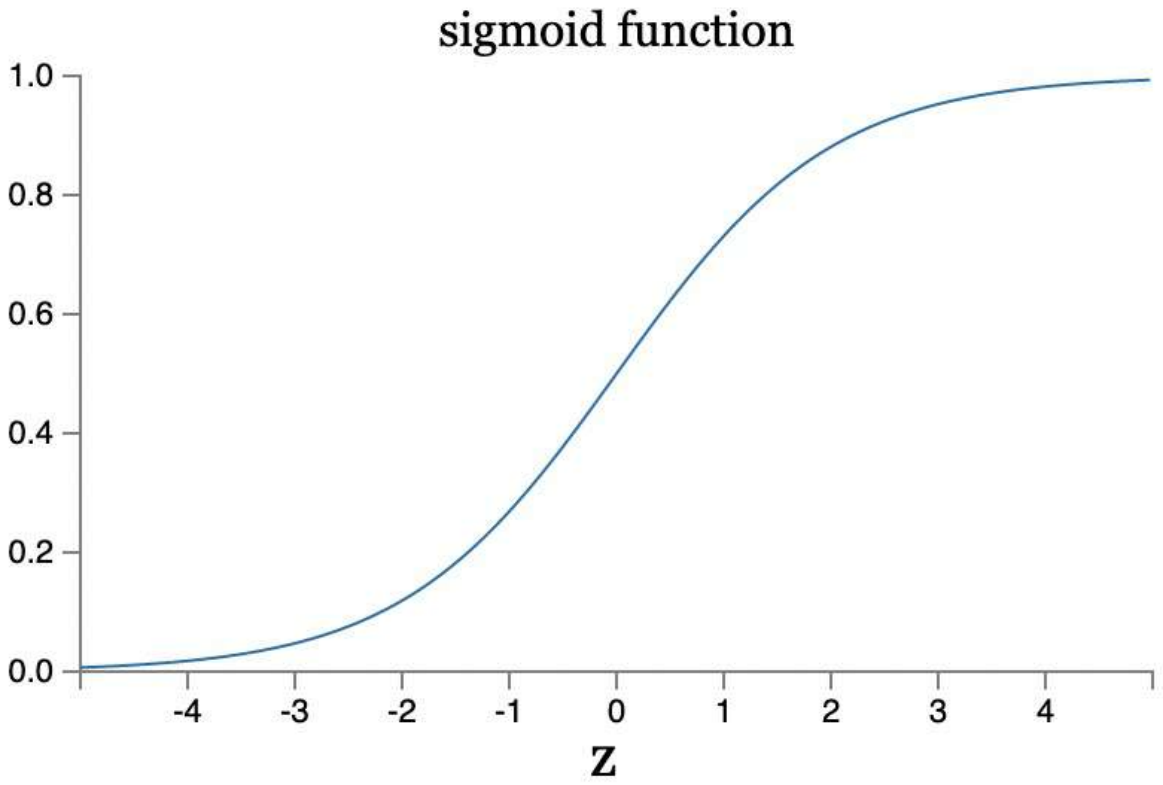
- Il reste à établir:
- a) que les réseaux neuronaux sont capables de représenter toute fonction qu'il serait souhaitable de représenter
- b) qu'il existe une méthode générale d'apprentissage qui pourrait permettre à un réseau d'apprendre une telle fonction

LE THÉORÈME D'APPROXIMATION UNIVERSELLE

- Le théorème d'approximation universelle : pour toute fonction continue F, pour tout degré d'approximation, un réseau neuronal peut approximer F à ce degré.

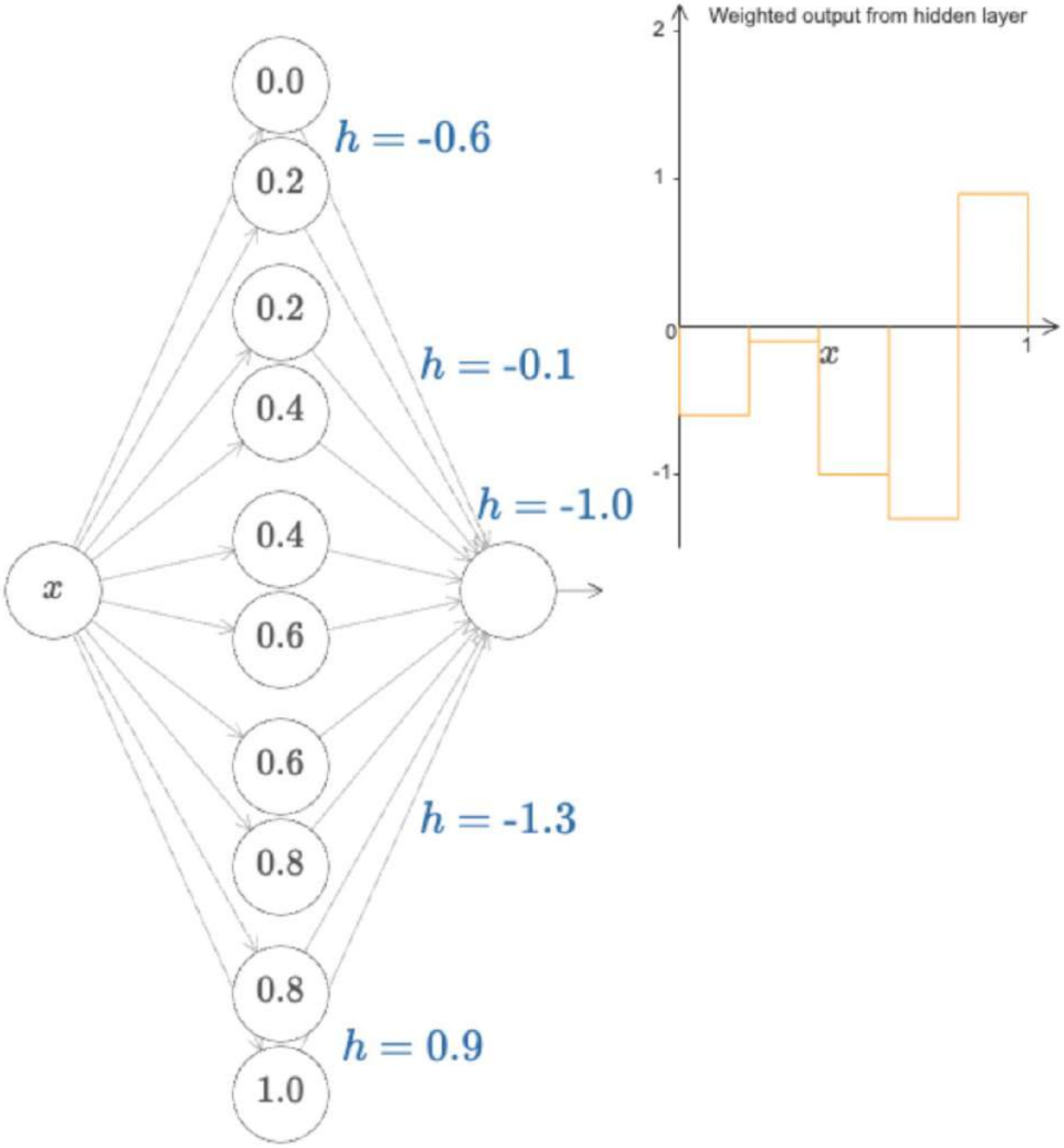
LE THÉORÈME D'APPROXIMATION UNIVERSELLE

- Comment ? Intuitivement, chaque neurone vous donne une courbe :


sigmoid function
1.0
0.8
0.6
0.4
0.2
0.0
-4
-3
-2
-1
0
1
2
3
4
Z

LE THÉORÈME D'APPROXIMATION UNIVERSELLE

- Vous pouvez ajuster la forme exacte de la courbe et l'endroit exact où elle se situe sur les axes x-y en ajustant les pondérations et le biais.
- Deux neurones, et vous avez deux courbes. Pour approximer une fonction arbitrairement courbe, vous avez juste besoin de suffisamment de neurones


0.0
0.2
0.2
0.4
0.4
0.6
0.6
0.8
0.8
1.0
x
h = -0.6
h = -0.1
h = -1.0
h = -1.3
h = 0.9
Weighted output from hidden layer
2
1
0
-1
x
1

LE THÉORÈME D'APPROXIMATION UNIVERSELLE

- Pour en savoir plus sur ce fonctionnement, consultez le chapitre :
- http://neuralnetworksanddeeplearning.com/chap4.html
- (les outils graphiques interactifs illustrent le propos bien mieux que je ne pourrais le faire)

LE THÉORÈME D'APPROXIMATION UNIVERSELLE

- Note : ce théorème s'applique même pour le cas d'un réseau avec une seule couche cachée avec beaucoup de neurones dans celle-ci entre les entrées et les sorties

DESCENTE DE GRADIENT ET BACKPROPAGATION

DESCENTE DE GRADIENT ET BACKPROPAGATION

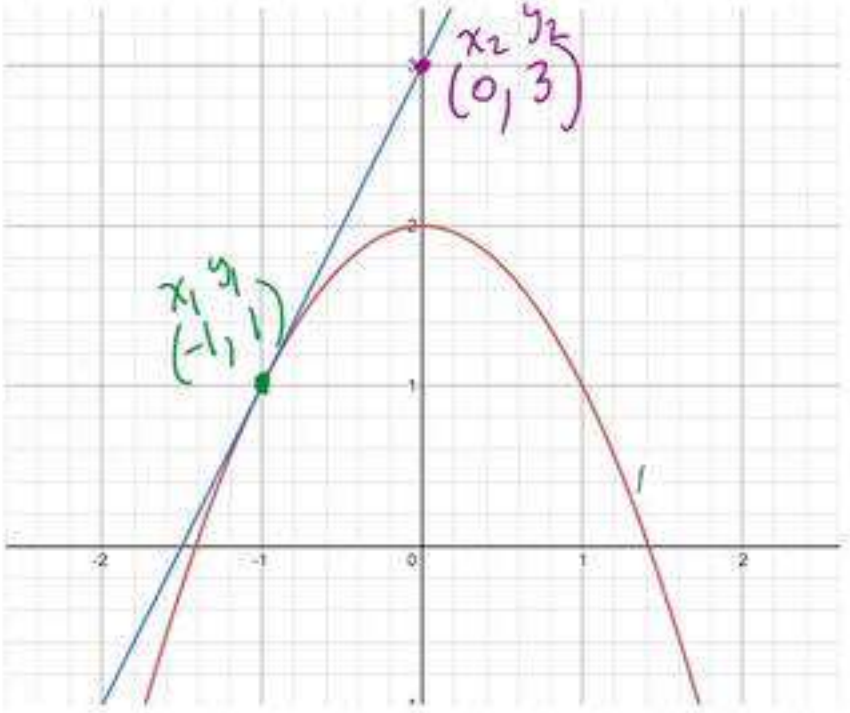
- J'ai mentionné qu'avec la régression linéaire, vous trouvez la ligne qui minimise l'erreur.

- Là, cela peut être fait analytiquement : vous pouvez en fait résoudre cette ligne (par exemple par la méthode des moindres carrés)

DESCENTE DE GRADIENT ET BACKPROPAGATION

- Dans les cas plus complexes, vous ne pouvez pas le faire de manière analytique.
- Cependant, vous pouvez identifier une "direction" dans laquelle chaque paramètre doit être légèrement modifié afin de réduire l'erreur.
- (l'approche, appelée rétropropagation / backpropagation, utilise des outils issus du calcul : la règle de la chaîne et les dérivées partielles, pour déterminer la "direction" dans laquelle l'erreur d'une fonction change lorsque vous la modifiez)

Ex. 1: Given the graph of f and its tangent line, estimate $f'(-1)$.



$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

$f'(-1) = m$

$m = \frac{3 - 1}{0 - -1} = \frac{2}{1}$

$m = 2$

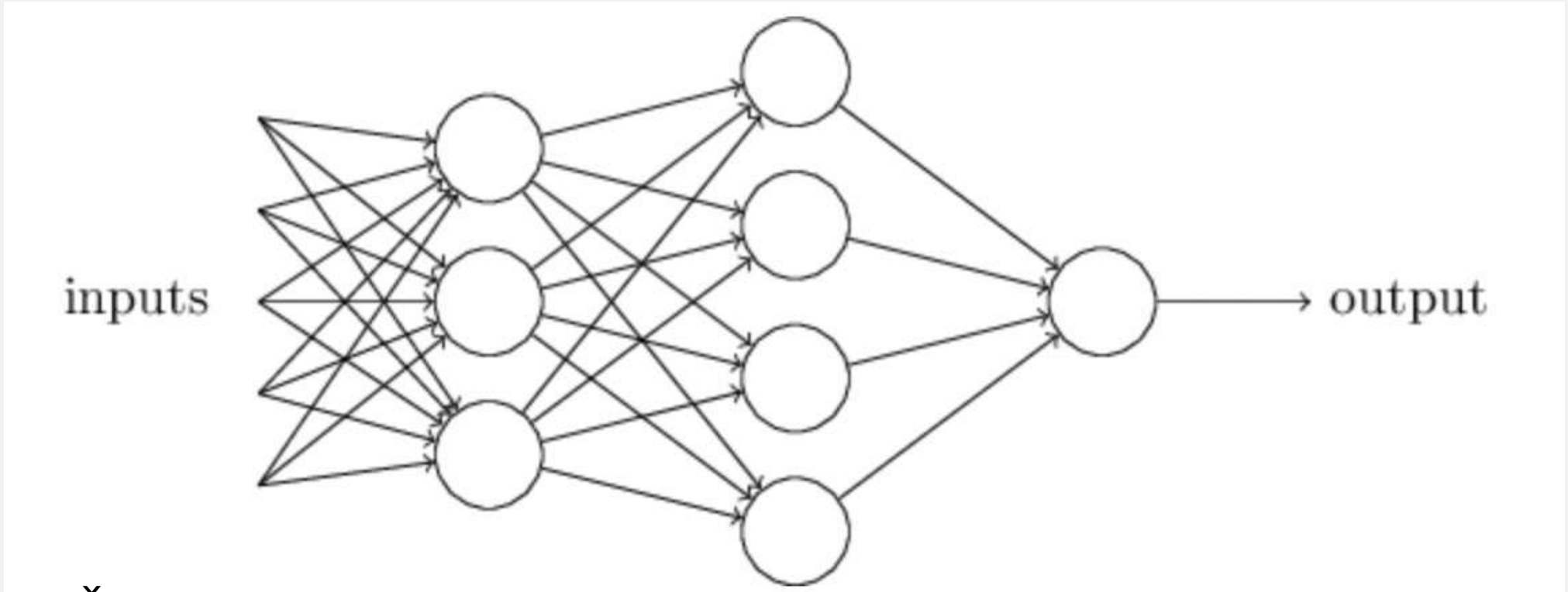
$f'(-1) = 2$

DESCENTE DE GRADIENT ET BACKPROPAGATION

- Mais ici en général c'est pour une fonction compliquée, multivariée, la fonction de coût de la fonction du réseau neuronal.

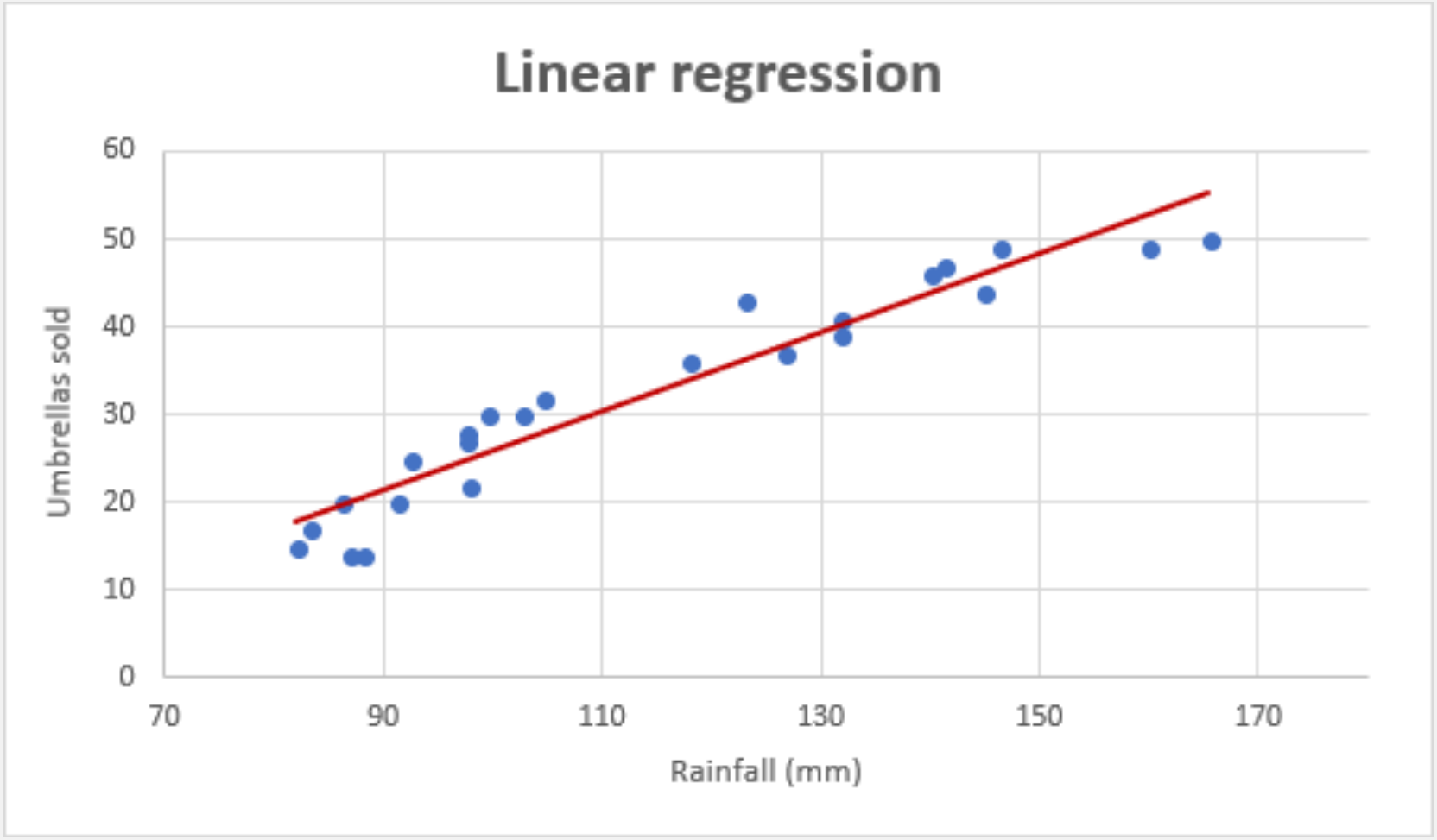
DESCENTE DE GRADIENT ET BACKPROPAGATION

- la fonction du réseau neuronal = F(G(H(x)))


inputs
output
x
H(X)
G(H(x)
F(G(H(x)))

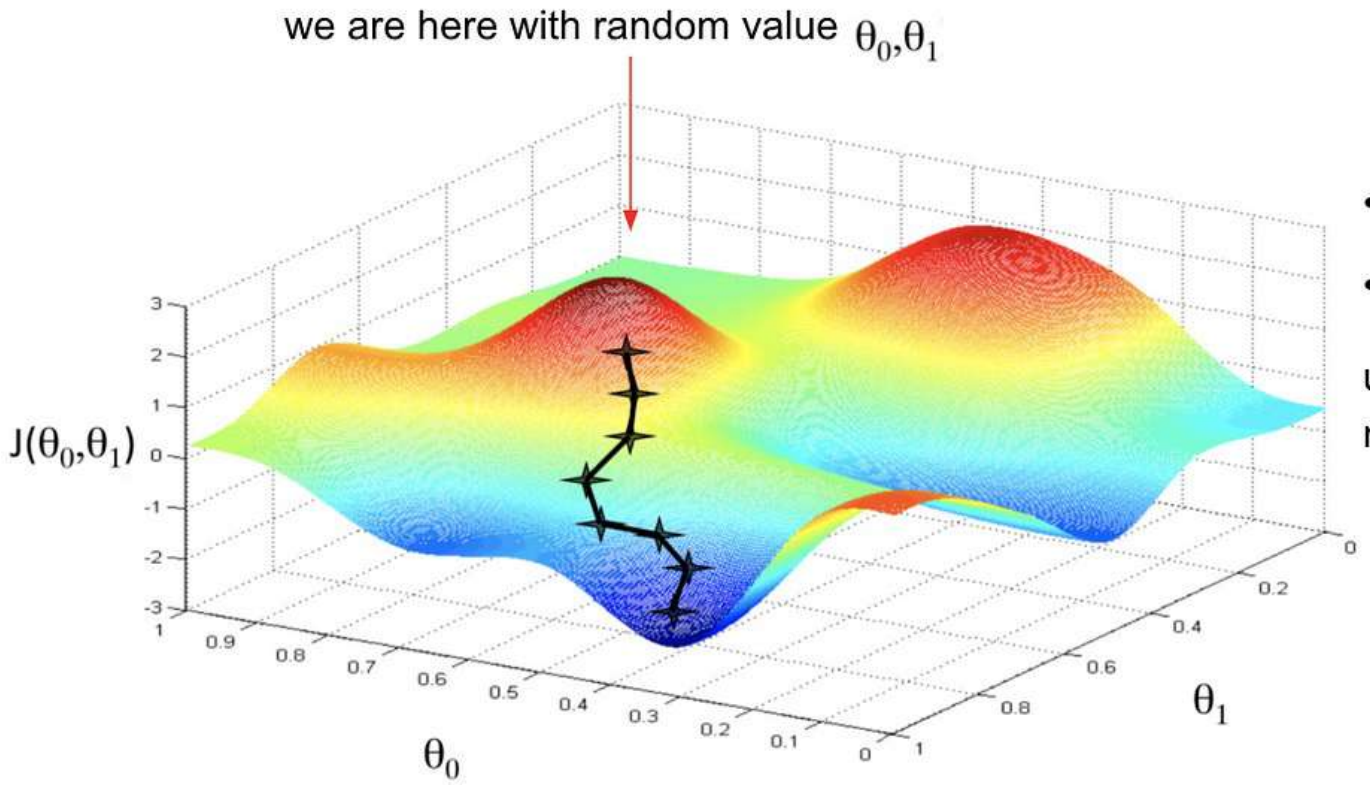
DESCENTE DE GRADIENT ET BACKPROPAGATION

- la fonction du réseau neuronal = F(G(H(x)))
- Sa fonction de coût C mesure la distance qui le sépare de la bonne réponse, pour chaque entrée.


Linear regression
Umbrellas sold
60
50
40
30
20
10
0
70
90
110
130
150
170
Rainfall (mm)

DESCENTE DE GRADIENT ET BACKPROPAGATION

- la fonction du réseau neuronal = F(G(H(x)))
- Sa fonction de coût C mesure la distance qui le sépare de la bonne réponse, pour chaque entrée.
- L'idée est de prendre la dérivée de
- C (F(G(H(x)))) :

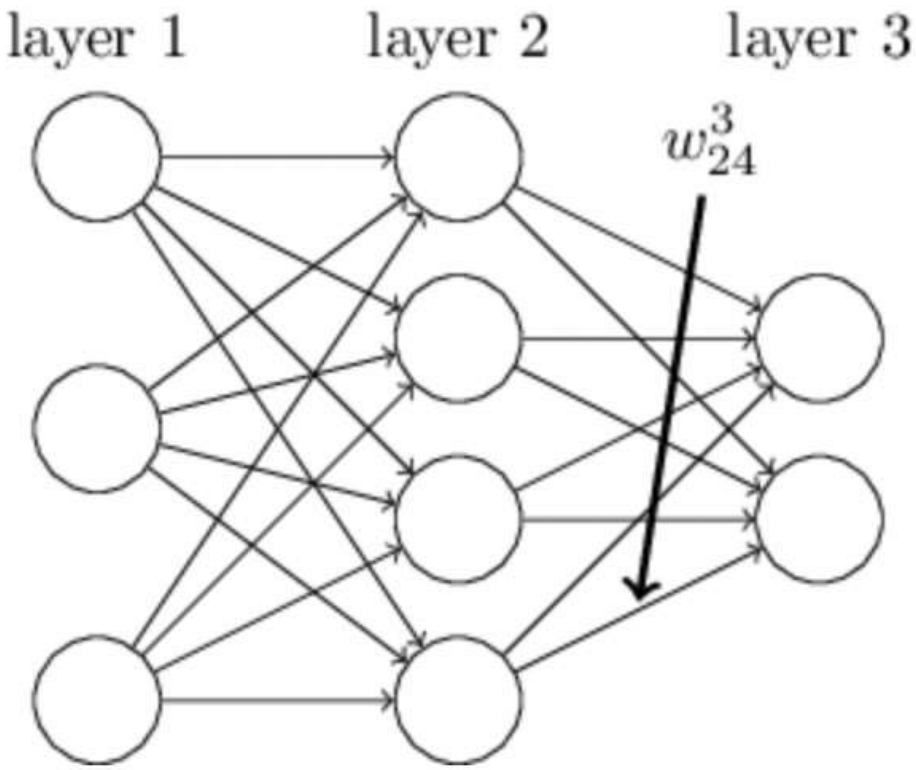


- Start with some θ_0,θ_1
- Keep changing θ_0,θ_1 to reduce $J(\theta_0,\theta_1)$

until we hopefully end up at a minimum

DESCENTE DE GRADIENT ET BACKPROPAGATION

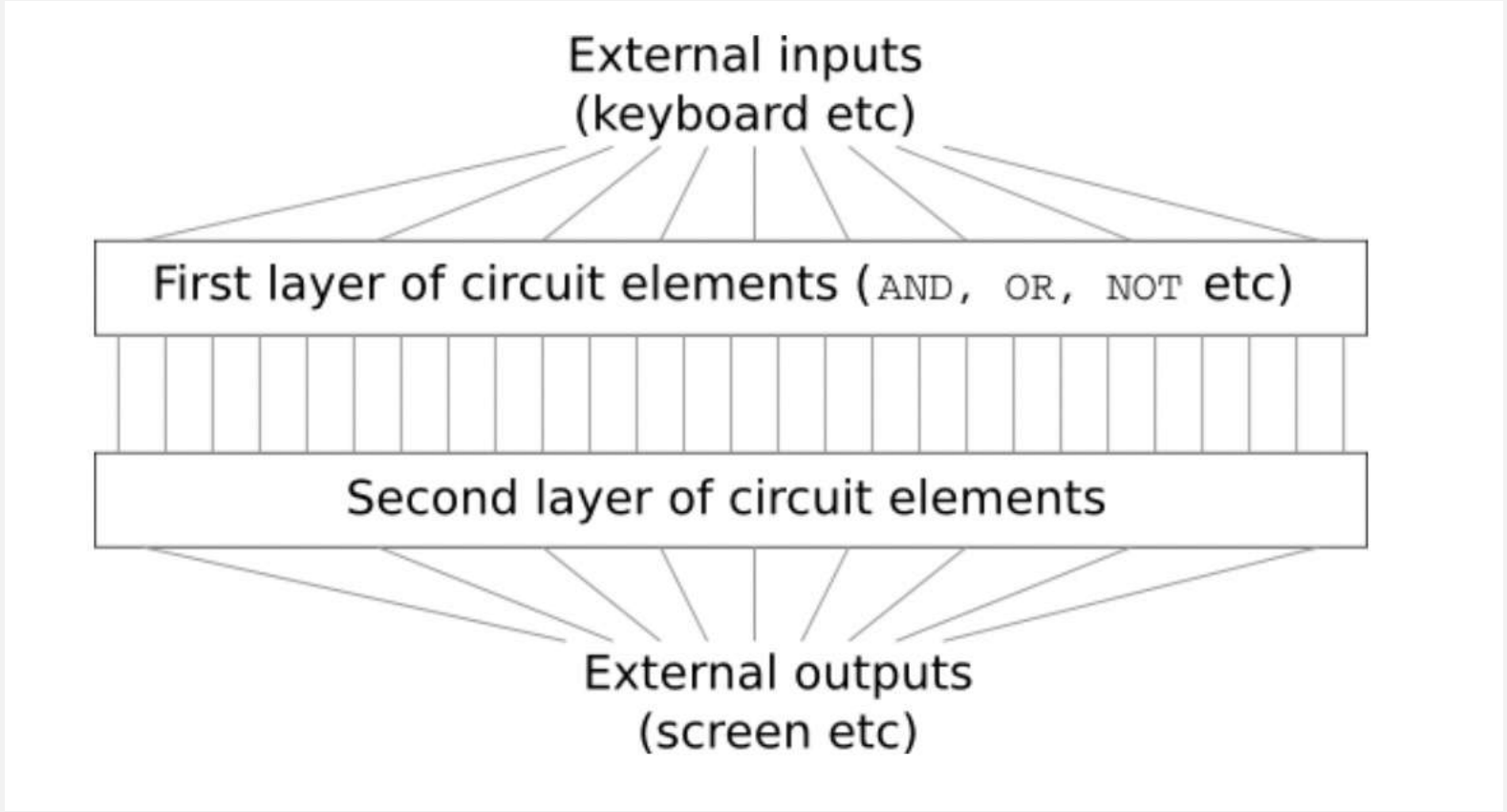
- la fonction du réseau neuronal = F(G(H(x)))
- Sa fonction de coût C mesure la distance qui le sépare de la bonne réponse, pour chaque entrée.
- L'idée est de prendre la dérivée de
- C (F(G(H(x)))) :
- et ensuite ajuster contre la direction de cette dérivée, c'est-à-dire dans la direction qui pointe vers le bas vers l'erreur minimale.

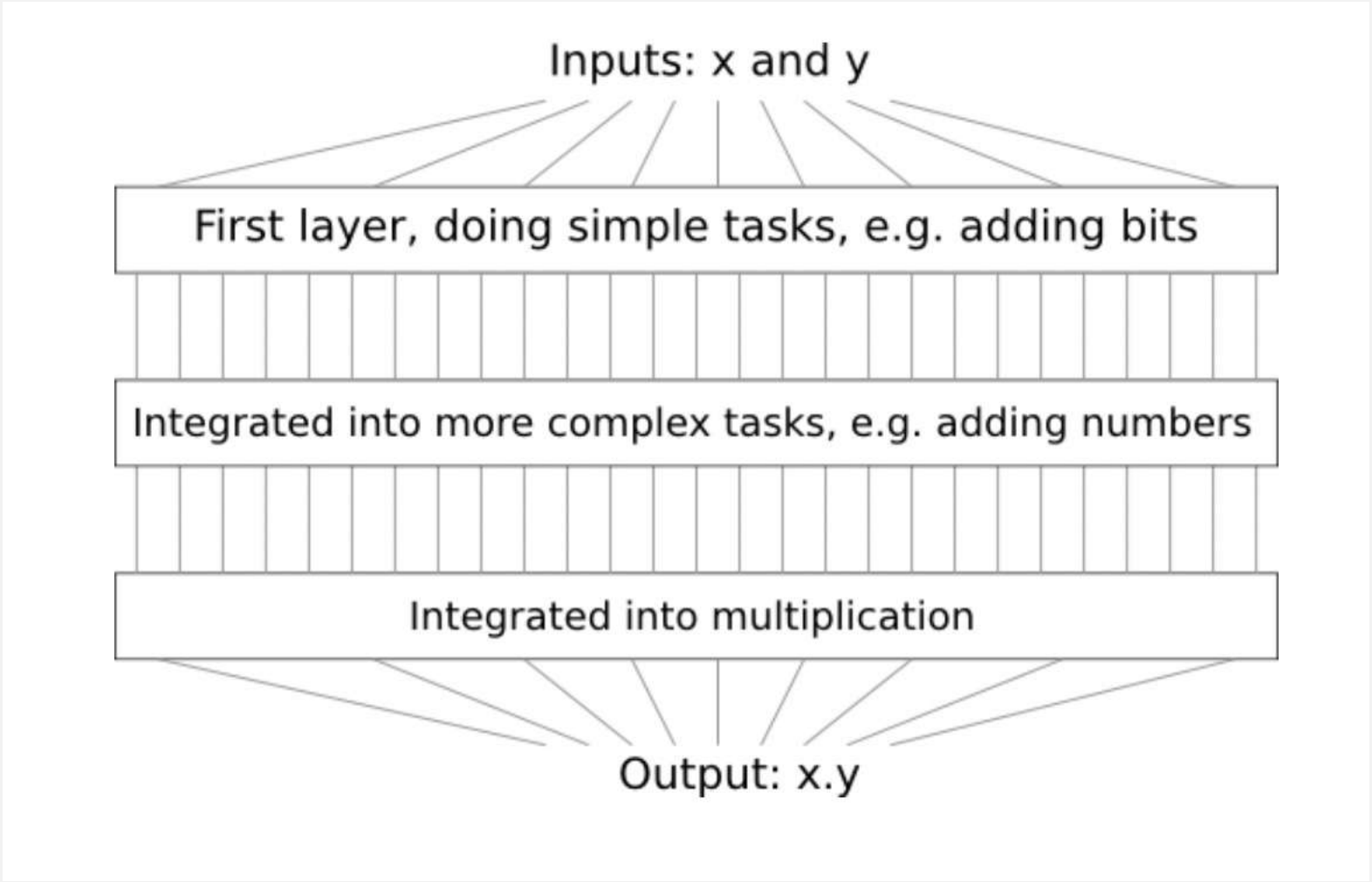


w^l_{jk} is the weight from the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer

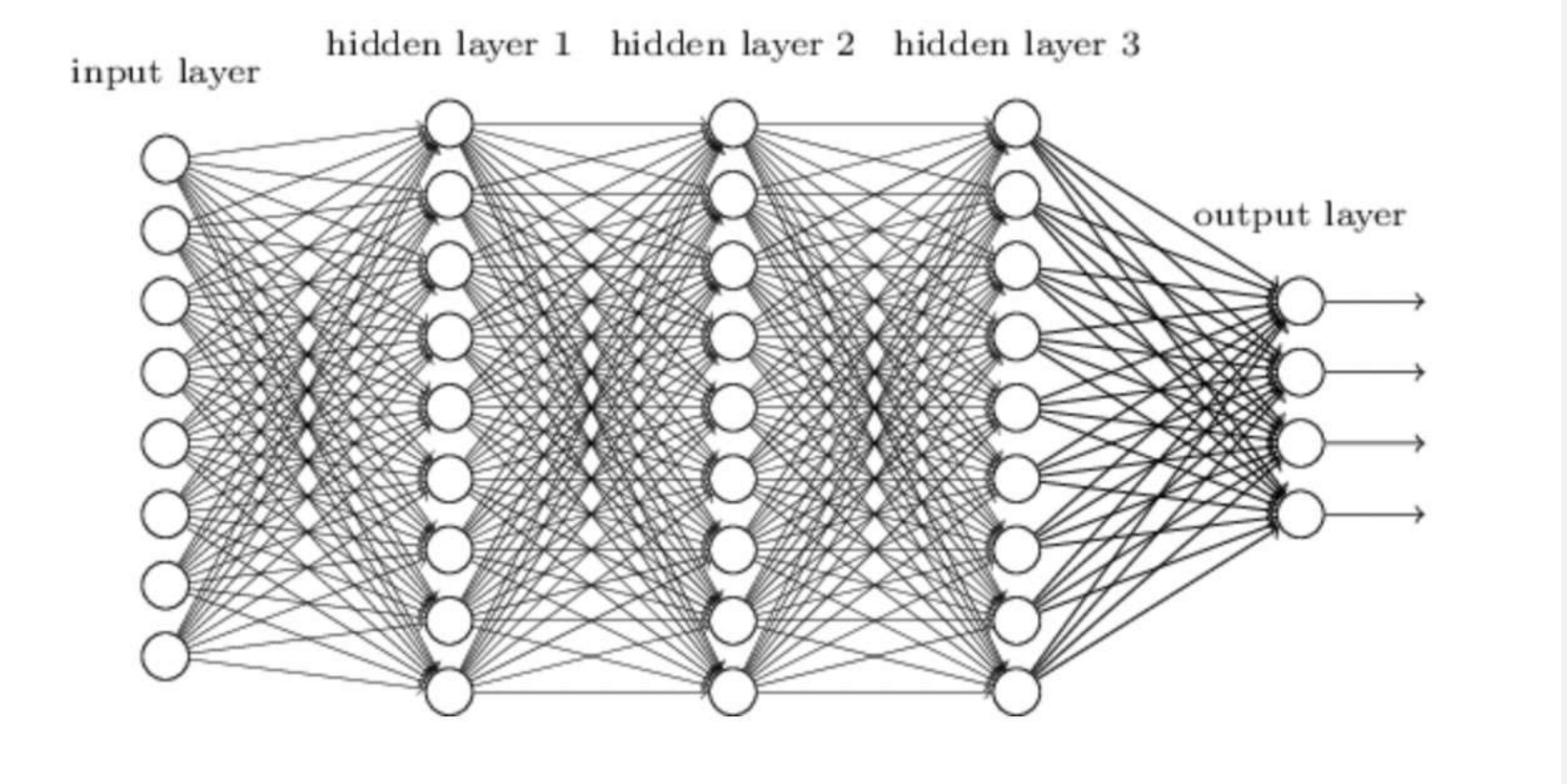
POURQUOI C'EST DIFFICILE

- Il s'avère que certaines fonctions sont très difficiles à apprendre (même lorsque nous avons beaucoup de données) !
- Surtout lorsqu'il n'y a pas assez de couches : cela revient à demander à quelqu'un d'effectuer toutes les étapes d'une tâche à plusieurs étapes en une seule fois


External inputs
(keyboard etc)
First layer of circuit elements (AND, OR, NOT etc)
Second layer of circuit elements
External outputs
(screen etc)


Inputs: x and y
First layer, doing simple tasks, e.g. adding bits
Integrated into more complex tasks, e.g. adding numbers
Integrated into multiplication
Output: x.y

- Mais en fait, les difficultés demeurent même avec un réseau multicouche (parfois appelé perceptron multicouche même si les activations sont sigmoïdes)
- Par exemple, les problèmes de gradient évanescent et explosif.


input layer
hidden layer 1
hidden layer 2
hidden layer 3
output layer

- Le gradient peut être trop petit ou trop grand dans les premières couches - en particulier pour les réseaux où chaque neurone de chaque couche est connecté à chaque neurone de la couche suivante.

- (en fait, l'apprentissage est un problème d'"attribution de crédits" - tu veux savoir quel neurone est responsable des erreurs, et quels neurones font du bon travail, et tu veux ajuster seulement ceux qui en ont besoin. Mais parfois, lorsque le réseau est trop encombré, cela ne fonctionne pas).

- Des réseaux plus complexes tels que les CNN, les RNN et les transformateurs se sont développés pour résoudre ces problèmes.

- Ces réseaux imposent des contraintes spéciales sur la façon dont les poids de certains neurones sont liés aux poids d'autres neurones, et contraignent également quels neurones sont connectés à quels autres neurones (contrairement aux réseaux "vanille" dans lesquels chaque neurone d'une couche est connecté à tous les autres neurones de cette couche).

- Comme nous le verrons, ces bouts de structure supplémentaires correspondent à différentes hypothèses ou "biais inductifs" sur la structure des données / du monde
- (Nous ne parlons pas ici du *biais* d'un seul neurone, mais de l'ensemble du système.)

- En d'autres termes, la taille compte, mais l'architecture aussi, à la fois la profondeur en général, mais aussi des questions plus spécifiques sur la façon dont le réseau est conçu, que nous explorerons....

- C'est intéressant et pertinent, par exemple, pour le débat sur le nativisme. Les réseaux neuronaux sont-ils des ardoises vierges ?

- Cependant, la question de savoir si un problème que tu surmontes avec une architecture intelligente (un biais inductif spécifique / une connaissance innée) ne pourrait pas aussi être surmonté simplement avec plus de neurones et plus de données reste débattue. (the scaling debate)