Conception d'un algorithme de Machine Learning

- Algorithms
- Pre-concepts
- Neurone
- Reseaux de Neurones
- BackPropagation

Algorithmes

Ils existe différents types d'algorithmes utilises dans le Machine Learning, cela sont les plus utilises:

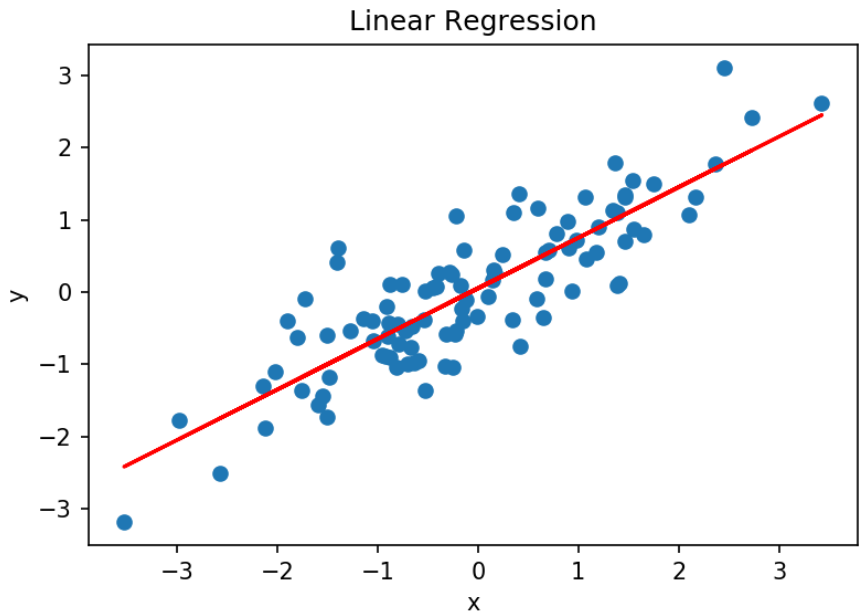
- Régression
- Réseaux Neuronal
- Clustering
- Arbre de régression
- Bayésien
- Réduction de dimension

Dans la suite on approfondira sur l'algorithme "Réseaux Neuronal".

Réseaux Neuronal

Pré-concept: Régression linéaire

Si ont ce rappels du cour de statistique, ont peut trace une droite la plus adapter, avec cet formule:



$$\hat{Y} = aX + b$$

Réseaux Neuronal

Pré-concept: Régression linéaire

Maintenant ont augmente les variables.

$\hat{Y} = b + w_0x_0 + w_1x_1 + w_2x_2 + ... + w_nx_n$

Cette formule est valable pour une seul donnée donc, pour notre nouage de points, on a ce tableaux:

$$\hat{Y}_0 = b + w_0x_0 + w_1x_1 + w_2x_2 + ...$$
$$\hat{Y}_1 = b + w_0x_0 + w_1x_1 + w_2x_2 + ...$$
$$\hat{Y}_2 = b + w_0x_0 + w_1x_1 + w_2x_2 + ...$$
$$.$$
$$.$$
$$\hat{Y}_m = b + w_0x_0 + w_1x_1 + w_2x_2 + ...$$

Tout ça on le passe a vecteurs, est notre formule ce transforme en ça:

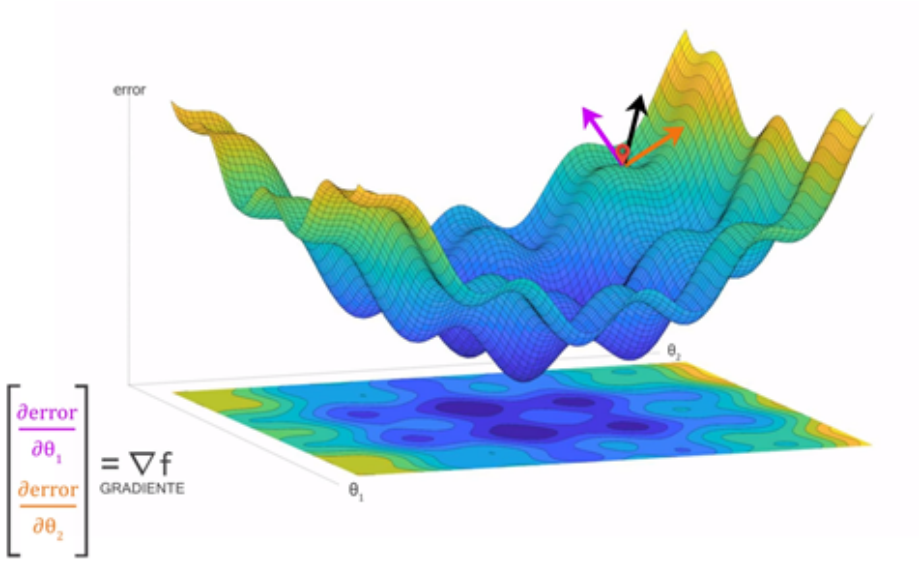
$$Y = WX$$

(Memoire de vietnam de algebre et analyse)

Réseaux Neuronal

Pré-concept: Gradient descent

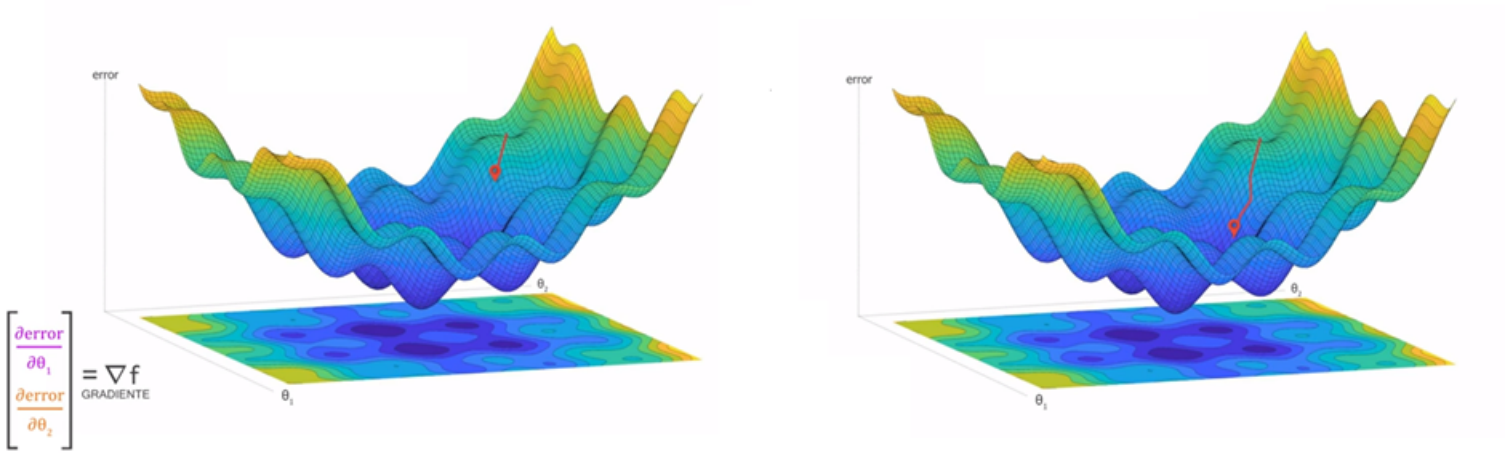
Ont peut pensée a un nuage de point comme un graph de profondeur, entre plus de points plus profond.



Si on utilise la dérivée partiel d'un point "P" ont trouve le vecteur de la pente ascendante "G"(le Gradient).

Réseaux Neuronal

Pré-concept: Gradient descent



//loop

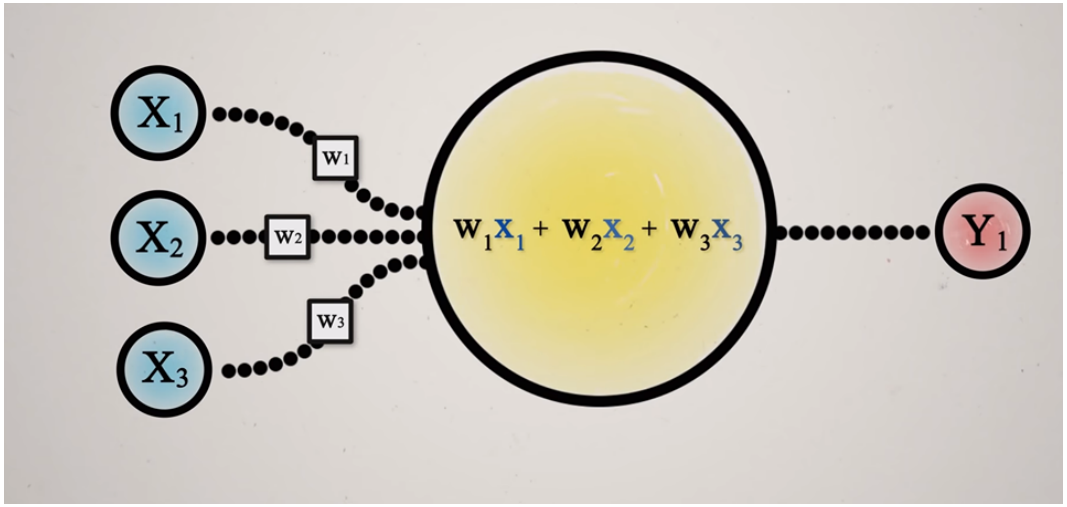
$$P_n = P_{n-1} - aG$$

Donc si ont l'inverse on peut trouver le vecteur descendent, a ça on le multiplie par une constante "a" que sera la distance de déplacement est ont répété de maniéré itérative.

Réseaux Neuronal

Neurone

Les neurones reçoit un vecteur de variables($x_0, x_1, .., x_n$), pour chaque variable ont a respectivement sont poids($w_0, w_1, .., w_n$), qui produis "Y".

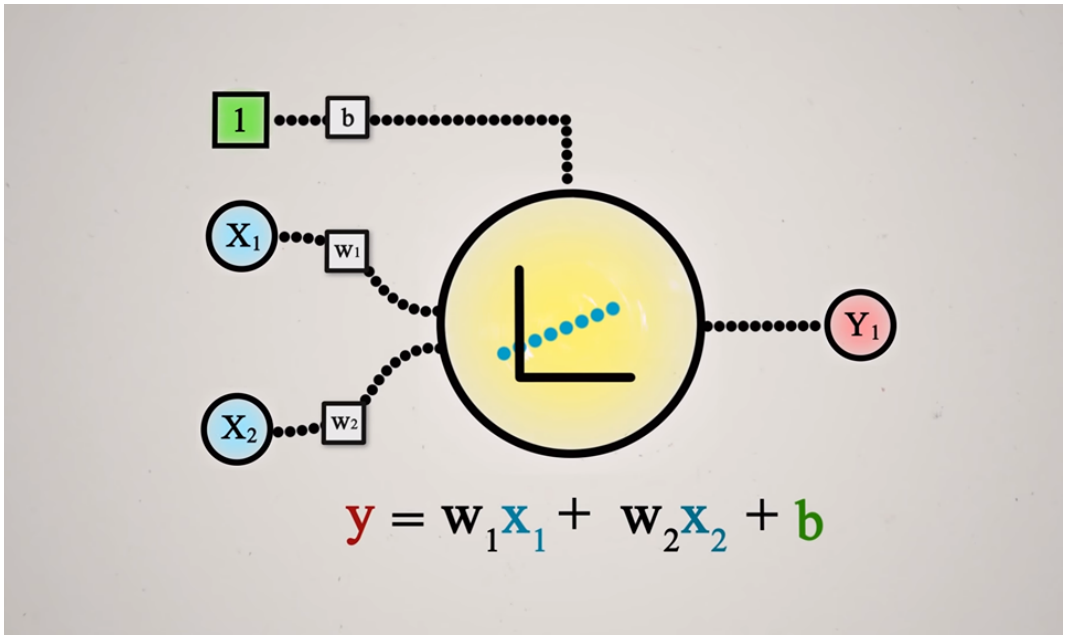


$$Y = w_0x_0 + w_1x_1 + .. + w_nx_n$$

Réseaux Neuronal

Neurone

A ce moment on ce rend compte que on a une equation linaire mais pour que ce soit complet on doit additionnée une constante "b" (bias) qui nous permettra modifier l'équation.



$$Y = b + w_0x_0 + w_1x_1 + .. + w_nx_n$$

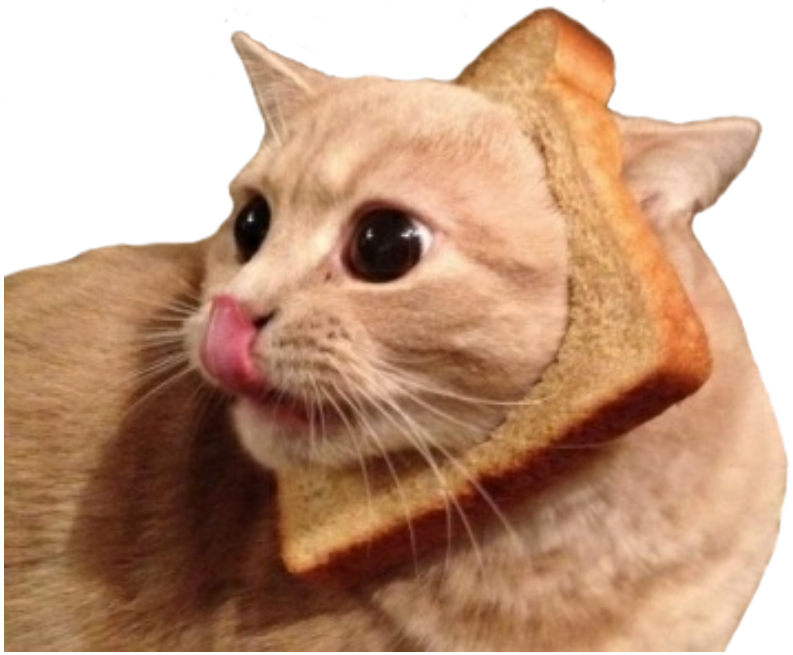
EXAMPLE!!!

Réseaux Neuronal

Neurone: EXAMPLE!!!!

Pour être Joyeux quand on est sur l'ordinateur on doit avoir 2 chose:

- Des images de chat avec du pan.
- Des equation différentiel de second degrés.



$$y'' = y' + 6y$$

Avec ces information on peut construire un Tableau de vérité.

TABLEAU DE VÉRITÉ

x_0	x_1	y
1	1	1
1	0	0
0	1	0
0	0	0

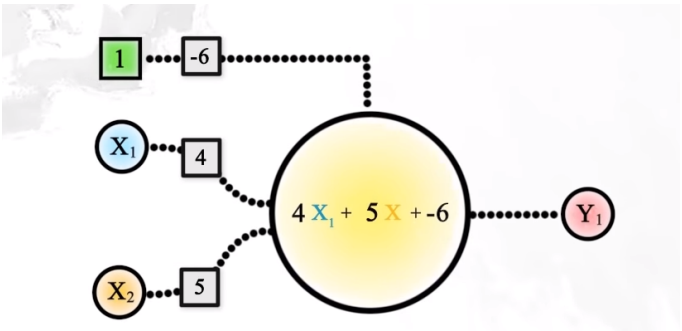
On doit définir une fonction de coste "C(X)", pour que la solution "Y" est pour résulta soit 1 ou 0.

XW - b $\geq$ 0 => 1
XW - b $<$ 0 => 0

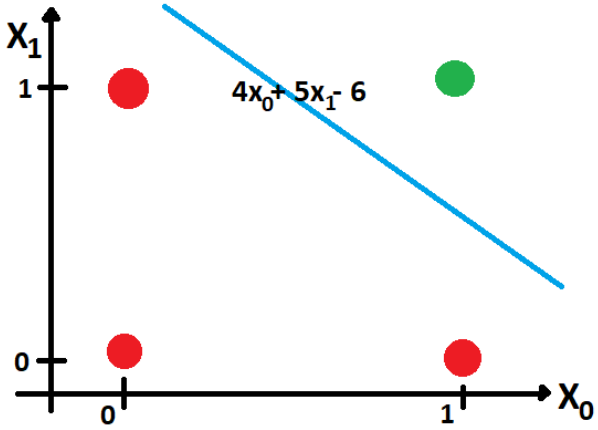
Réseaux Neuronal

Neurone: EXAMPLE!!!!

A partir de la on fait tourner la machine jusqu'à que le tableau de vérité soit correct.



$x_0 = 1$ & $x_1 = 1$
$4(1) + 5(1) - 6 = 3 \Rightarrow C(3) = 1$
$x_0 = 1$ & $x_1 = 1$
$4(1) + 5(0) - 6 = -2 \Rightarrow C(-2) = 0$
$x_0 = 1$ & $x_1 = 1$
$4(0) + 5(1) - 6 = -1 \Rightarrow C(-1) = 0$
$x_0 = 1$ & $x_1 = 1$
$4(0) + 5(0) - 6 = -6 \Rightarrow C(-6) = 0$

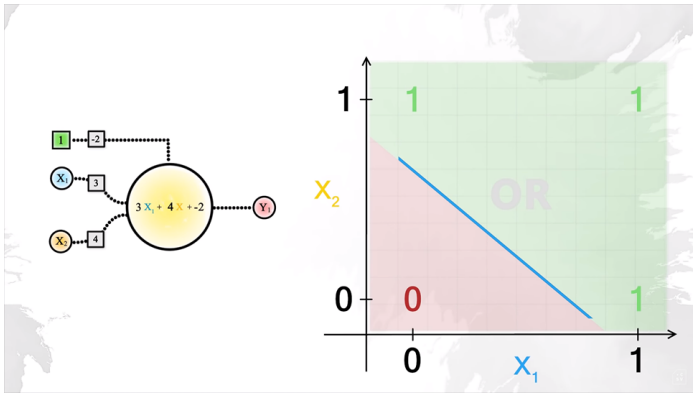


"ET" logique?

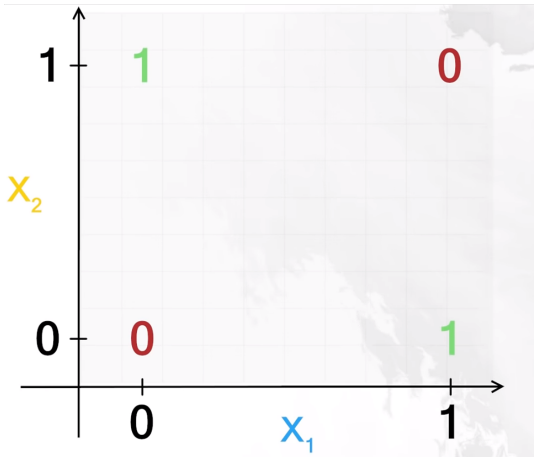
Réseaux Neuronal

Neurone: EXAMPLE!!!!

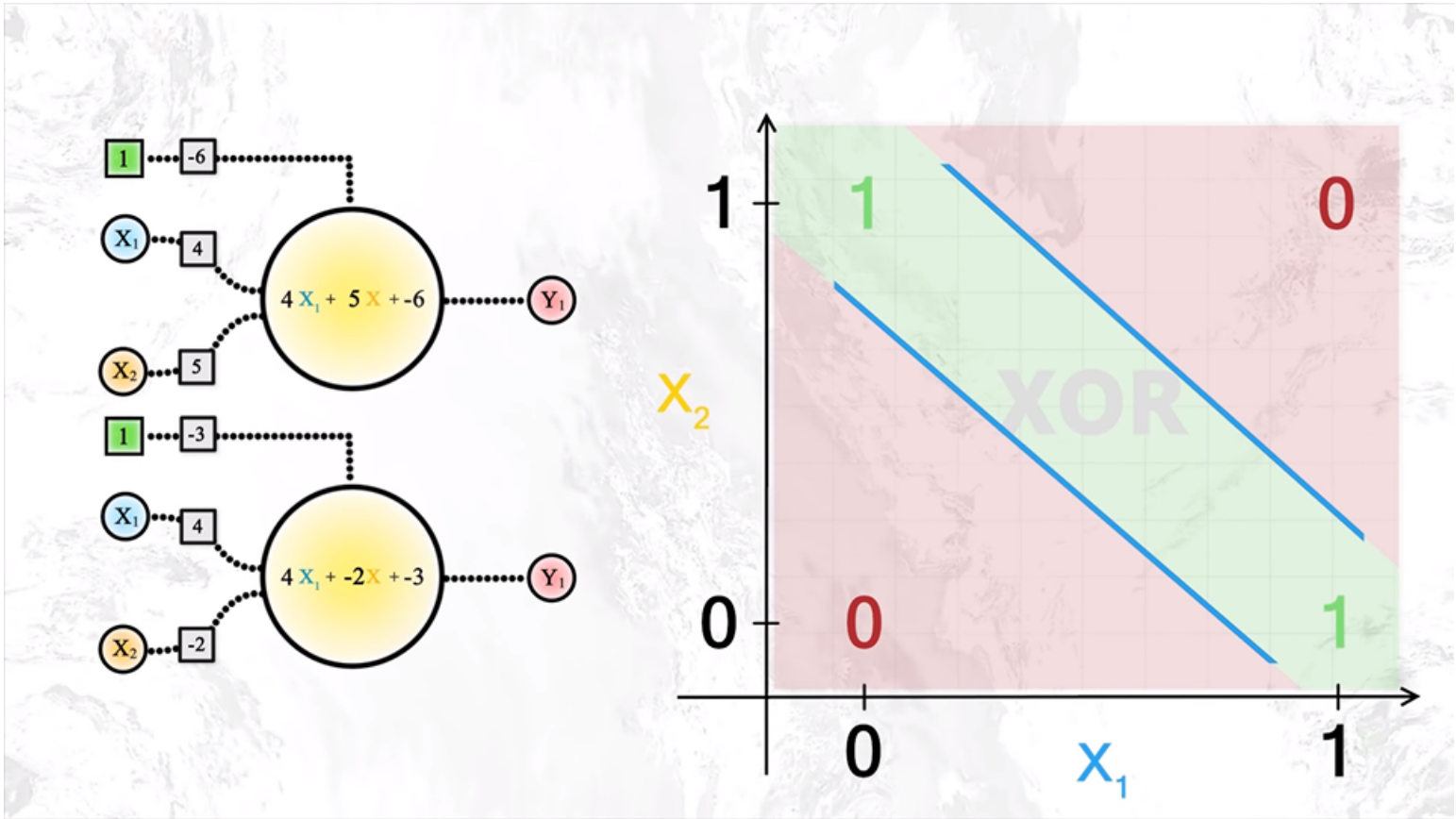
"OU" logique.



"Ou exclusif" comment le résoudre?



Multi Neurone

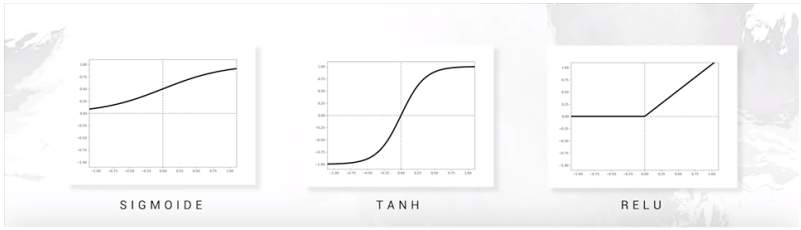


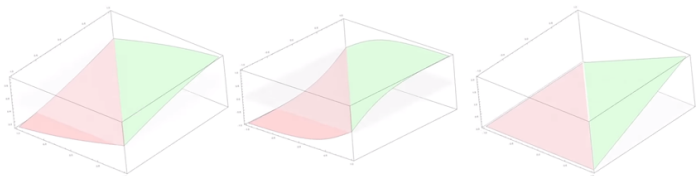
Pour travailler avec un Réseaux ont doit les organiser. On les organise par "cape" (n "cape"), chaque "cape" aura m neurones, la premier "cape" (C_0 "cape" d'entrée) est la dernier (C_n cape de sortie), tous les capes intermédiaire serons appelée des "capes" cachées ($C_0 < C_p < C_n$). La cape C_p reçoit des valeur de la cape C_{p-1} est envoi ces résultats a la cape C_{p+1}.

PROBLEM!!

si on addition n fonction linéal on a comme résulta une fonction linéal que ce comme si on aure que une, donc on doit faire apparaître le concept de "Fonction d'activation". Ces fonction modifira notre fonction linéal en no-linéal:

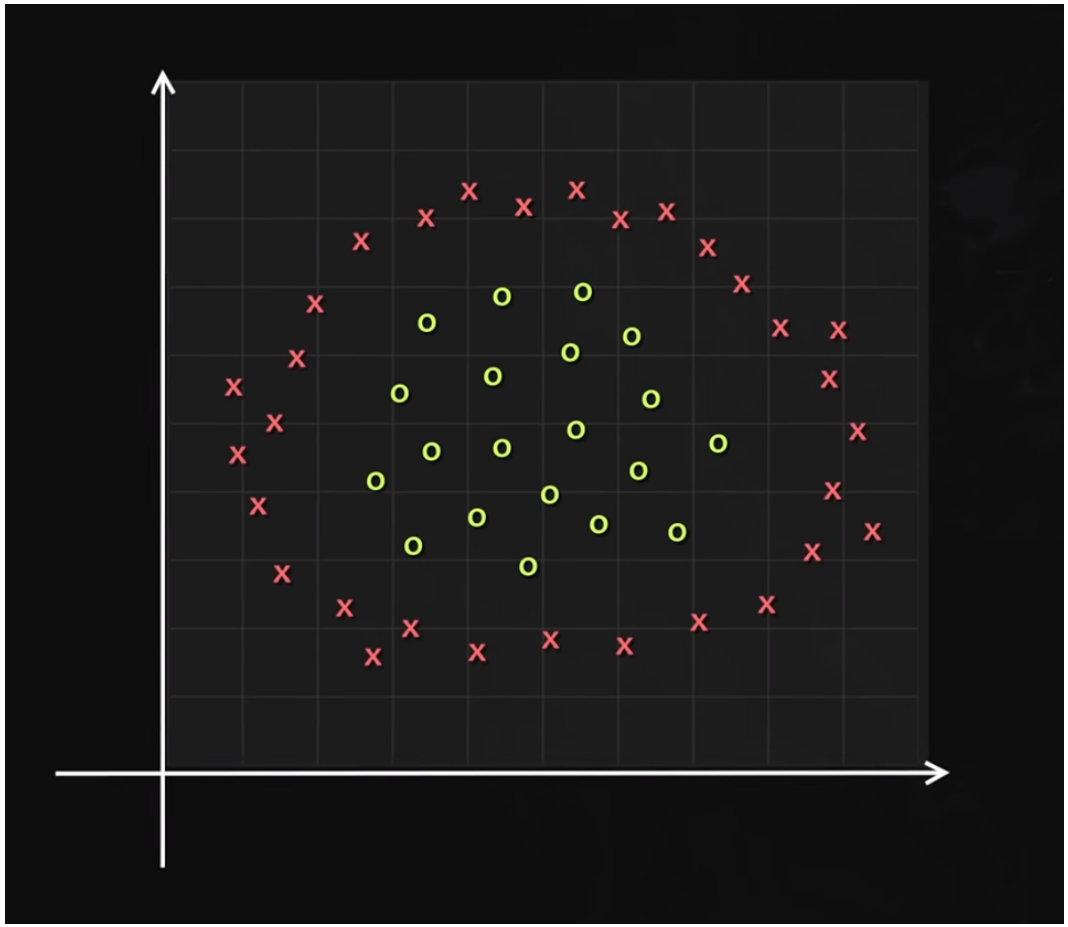
- RELU
- TANH
- SIGMOIDE



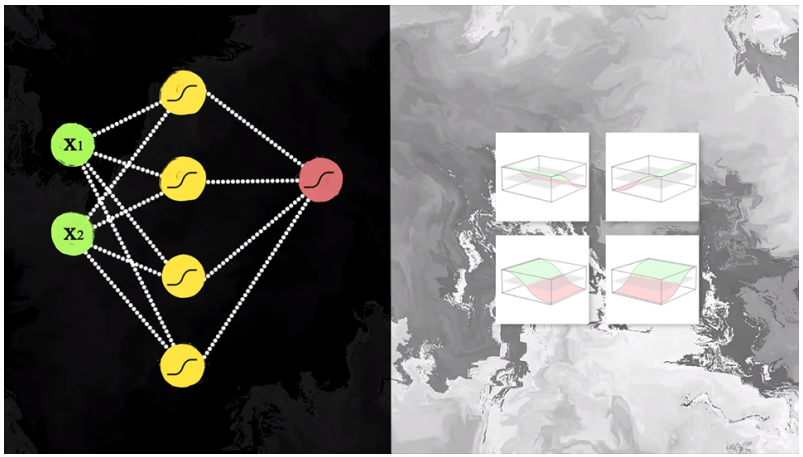


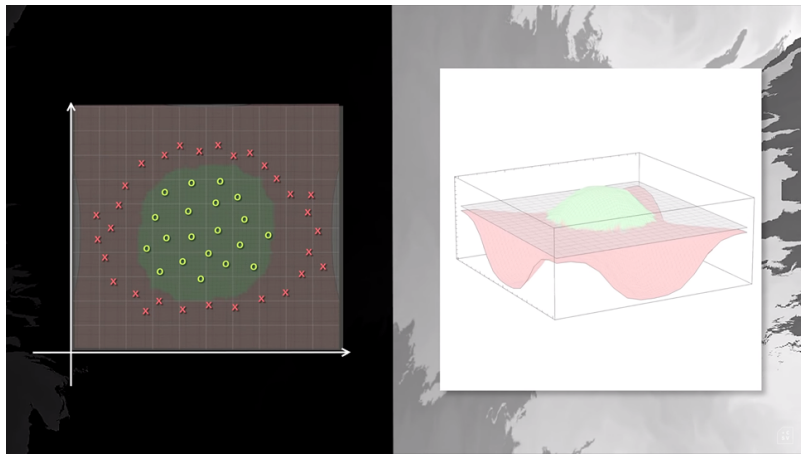
sont des fonction courant.

Pour résoudre un problem de ce type:



On peut imbrique de fonction sigmoide comme ça:





Maintenant on doit pouvoir faire que notre algorithme peux apprendre, par lui mème, ces équivalent a dire que l'algorithme trouve la droite la plus optimise.
Donc on doit appliqué le Gradient descent a chaque neurone de la fin au debut. Pour ça on définit les fonction utiliser dans les neurones.

$Z =>$ la somme des variables et le bias $(w_0 * x_0 + .. + w_n * x_n + b)$.
A(x) => la fonction d'activation.
C(x) => la fonction de coste.
On les imbriques:

$$C(A(Z^n)) => C(A(w_0 * x_0 + .. + w_n * x_n + b))$$

et a chaque neurone de chaque cape(L capes) on le fait les dériver partielle.

$$\frac{\partial C}{\partial w^L} \qquad \frac{\partial C}{\partial b^L}$$

PLUS DE MATHEMATIQUE!!!!

$$\frac{\partial C}{\partial w^L} = \frac{\partial C}{\partial A^L} * \frac{\partial A^L}{\partial Z^L} * \frac{\partial Z^L}{\partial w^L}$$

$$\frac{\partial C}{\partial b^L} = \frac{\partial C}{\partial A^L} * \frac{\partial A^L}{\partial Z^L} * \frac{\partial Z^L}{\partial b^L}$$

$$\frac{\partial Z^L}{\partial b^L} = 1 \qquad \frac{\partial Z^L}{\partial w^L} = Y^{L-1} \qquad \frac{\partial C}{\partial A^L} * \frac{\partial A^L}{\partial Z^L} = \delta^L$$

$\frac{\partial C}{\partial A^L} =>$ la dérive de la fonction de coste.
$\frac{\partial A^L}{\partial Z^L} =>$ la dérive de la fonction d'activation.

on remplace dans l'équation

$$\frac{\partial C}{\partial w^L} => \delta^L Y^{L-1}$$

$$\frac{\partial C}{\partial b^L} => \delta^L$$

tout ça uniquement pour la dernier cape!!!!

Maintenant en propage aux autre capes.

$$\delta^L = \frac{\partial C}{\partial A^L} * \frac{\partial A^L}{\partial Z^L}$$

$$\delta^{L-1} = w^L \delta^L * \frac{\partial A^{L-1}}{\partial Z^{L-1}}$$

$$\frac{\partial C}{\partial b^{L-1}} = \delta^{L-1} \qquad \frac{\partial C}{\partial w^{L-1}} = \delta^{L-1} * Y^{L-2}$$

VOILA!!

Maintenant on peut trouver la cure au cancer Lite et en aspirine.