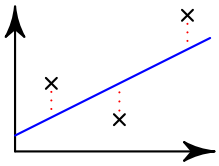
Régression linéaire

Intelligence artificielle

Il existe différentes techniques de programmation qui rentrent dans la catégorie **intelligence articielle**. Le but est de simuler une forme d'intelligence. Cela va d'algorithmes très simples pour les adversaires dans les jeux vidéo, jusqu'à des algorithmes très poussés capables de remplacer un visage par un autre dans une vidéo. Essentiellement, on peut voir une intelligence artificielle comme une fonction prenant certaines données et devant produire certains résultats. Selon la complexité du problème, cette fonction est "facilement" programmable ou au contraire, il semble compliqué de mettre au point une telle fonction.
L'utilisation de **réseaux de neurones** a permis de faire d'énormes avancées. Ce sont des fonctions reposant sur un très grand nombre de paramètres. Afin de trouver les bonnes valeurs pour ces paramètres, on utilise des **techniques d'apprentissage**, telles que les algorithmes génétiques et **l'apprentissage profond**. Cette dernière approche est basée sur la régression linéaire.

Régression linéaire

Imaginons que l'on dispose d'un ensemble de points, appelé **nuage de points** dans un repère et qu'ils ont l'air plus ou moins alignés. On peut alors se demander quelle est la droite qui passe *au mieux* entre ces points. Nous allons utiliser la **méthode des moindres carrés** pour trouver les coefficients de cette droite.



Plus formellement, étant donné un ensemble de n points $\{(x_1,y_1),\ldots,(x_n,y_n)\}$, on recherche les coefficients a et b tels que la droite $y = ax + b$, appelée **droite de régression**, approxime au mieux le nuage de points. Pour cela, elle doit minimiser la fonction de coût $C(a,b) = \sum_{i=1}^{n} (ax_i + b - y_i)^2$.
On calcule le carré de l'erreur pour chaque x_i entre le y calculé et y_i et on cherche la droite qui a le coût minimal.

Approche par tâtonnement

Prenons comme exemple les points $\{(0,0),(1,1),(2,1)\}$. Afin de déterminer les deux coefficients de la droite de régression, on peut essayer quelques valeurs, à l'aide d'un tableau :

Coeffs	x_i	ax_i+b	y_i	$(ax_i+b-y_i)^2$
a=0 b=0	0	0	0	0
	1	0	1	1
	2	0	1	1
			$C(a,b)=$	2

Coeffs	x_i	ax_i+b	y_i	$(ax_i+b-y_i)^2$
a=1 b=0	0		0	
	1		1	
	2		1	
			$C(a,b)=$	

Coeffs	x_i	ax_i+b	y_i	$(ax_i+b-y_i)^2$
a=0 b=1	0		0	
	1		1	
	2		1	
			$C(a,b)=$	

Coeffs	x_i	ax_i+b	y_i	$(ax_i+b-y_i)^2$
a=0,5 b=0	0		0	
	1		1	
	2		1	
			$C(a,b)=$	

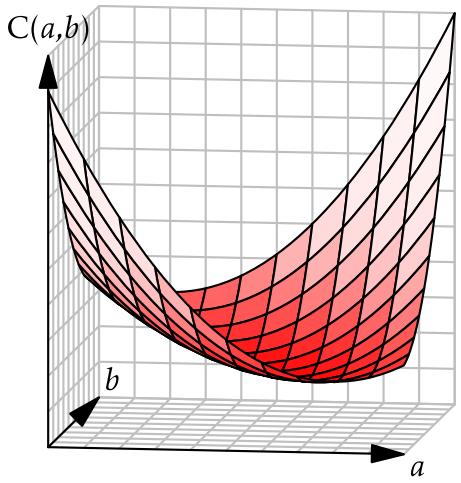
Exercice 1 : Compléter les trois autres tableaux et déterminer, parmi celles proposées, quelles valeurs de a et de b donnent le plus petit coût.

Approche géométrique

Afin de minimiser C(a,b), on peut essayer de trouver la formule explicite en remplaçant les x_i et y_i par leurs valeurs :

$$\begin{aligned}
\mathrm{C}(a,b) &= (a\times 0+b-0)^2+(a\times 1+b-1)^2+(a\times 2+b-1)^2\\
&= b^2+(a+b-1)^2+(2a+b-1)^2\\
&= b^2+a^2+b^2+1+2ab-2a-2b+4a^2+b^2+1+4ab-4a-2b\\
&= 5a^2+6ab-6a+3b^2-4b+2
\end{aligned}$$

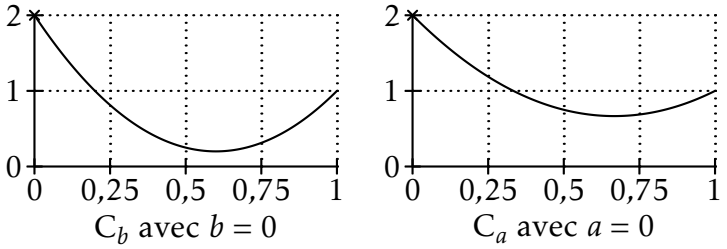
La fonction C est une fonction à deux variables, qui peut être représentée par une surface en 3 dimensions, comme ci-contre. Minimiser cette fonction revient donc à trouver les coordonnées du point le plus bas de la surface.



Pour cela, on peut étudier les fonctions $\mathrm{C}_b : a \mapsto \mathrm{C}(a,b)$ et $\mathrm{C}_a : b \mapsto \mathrm{C}(a,b)$. Pour une valeur donnée de b, on peut tracer la courbe de C_b et de même pour une valeur donnée de a et C_a. Cela correspond aux courbes tracées sur la surface ci-contre.

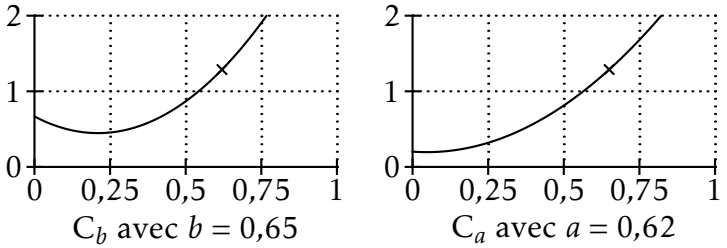
La figure ci-dessous montre les deux courbes avec $a = 0$ et $b = 0$. Le croix indique en quel point les deux courbes se coupent sur la surface.

On peut remarquer que l'on se trouve au point de coordonnées (0,0,2) sur la courbe et qu'il est possible de descendre plus bas en augmentant les deux paramètres.



C_b avec $b = 0$ $\quad$ C_a avec $a = 0$

Le minimum de chaque courbe semble obtenu avec le couple (0.62,0.65). On obtient alors les 2 courbes ci-contre. Le résultat n'est pas vraiment celui attendu. Le point se trouve moins haut, mais pas autant que prévu.



C_b avec $b = 0{,}65$ $\quad$ C_a avec $a = 0{,}62$

Cela vient du fait que lorsqu'on modifie la valeur de a, on change également C_a et de même pour b et C_b. Il ne faut donc pas voir les courbes commes indépendantes, mais plutôt s'en servir pour déterminer s'il vaut mieux modifier beaucoup ou peu chaque paramètre. Pour trouver le minimum d'une fonction, on peut imaginer lâcher une bille sur la courbe et la laisser rouler jusqu'à ce qu'elle se stabilise dans un creux (qui n'est pas forcément le minimum de la fonction). De la même manière, pour une fonction à deux variables, on peut lâcher une bille sur la surface correspondante. Il faut donc déterminer dans quelle direction va se déplacer la bille : plutôt selon l'axe de a, celui de b, ou autant l'un que l'autre. Pour déterminer cela, il faut regarder sur quelle courbe la pente est la plus importante.

Ainsi, en partant de (0,0), il faut augmenter plus la valeur de a que de b. On peut alors avancer un petit peu, regarder les deux pentes et recommencer jusqu'à arriver en bas.

Utilisation des dérivées

Afin de "calculer" de déplacement à faire sur chacun des deux axes, nous allons utiliser les fonctions dérivées C'_a et C'_b qui donnent les pentes de chaque courbe, en fonction de l'autre paramètre. On trouve $C'_b(a) = 10a + 6b - 6$ et $C'_a(b) = 6a + 6b - 4$.

Exercice 2 : On prend $a = 0$ et $b = 0$. Calculer $C'_b(0)$ et $C'_a(0)$.

On peut remarquer que les deux dérivées sont négatives, alors qu'il faut augmenter les deux valeurs. C'est parce que la dérivée est positive quand la courbe monte et négative quand la courbe descend. Il faut donc rajouter une valeur correspondant à l'opposé de la dérivée. Afin de ne pas se déplacer trop loin et risquer de dépasser et de s'éloigner encore plus du minimum, on multiplie les dérivées par un paramètre λ, que l'on appelle **le pas**. Les nouvelles valeurs de a et de b seront donc $a - \lambda C'_b(a)$ et $b - \lambda C'_a(b)$.

Exercice 3 : On part de $a = 0$ et $b = 0$, avec un pas de $\lambda = 0{,}1$. Calculer les prochaines valeurs de a et de b.

En itérant ce processus, les valeurs finissent par converger et ne changent quasiment plus :

a	b	$C(a,b)$	$C'_b(a)$	$a - \lambda C'_b(a)$	$C'_a(b)$	$b - \lambda C'_a(b)$
0	0	2	6	0,6	4	0,4
0,6	0,4	0,52	2,4	0,36	2	0,2
0,36	0,2	0,24	−1,2	0,48	−0,64	0,264
0,48	0,264	0,18541	0,384	0,4416	0,464	0,2176
0,4416	0,2176	0,17366	−0,2784	0,46944	−0,0448	0,22208
0,46944	0,22208	0,17039	0,02688	0,46675	0,14912	0,20717
⋮	⋮	⋮	⋮	⋮	⋮	⋮
0,5	0,166667	0,166667	$5{,}6 \times 10^{-17}$	0,5	$2{,}2 \times 10^{-16}$	0,166667

Avec cette approche, on arrive à $a = 0{,}5$ et $b = \dfrac{1}{6}$ qui semblent optimales. On peut d'ailleurs remarquer qu'avec ces deux valeurs, les dérivées sont nulles, ce qui signifie qu'on a atteint un minimum local (ou un col, mais c'est hautement improbable).
Selon le pas choisi, la forme de la courbe et les valeurs de départ, ce processus peut diverger ou converger vers un minimum qui n'est pas le vrai minimum. Néanmoins, dans le cas d'un ajustement affine, il n'y a qu'un seul minimum, qu'on aurait pu trouver en résolvant le système $\begin{cases} C'_b(a) = 0 \\ C'_a(b) = 0 \end{cases}$.

Méthode directe

L'approche que nous venons d'utiliser peut devenir fastidieuse si le nombre de points est grand. La méthode des moindres carrés permet de calculer directement les coefficients. Il faut d'abord calculer $\overline{x}$ et $\overline{y}$ qui sont les valeurs moyennes des x_i et des y_i. On obtient ensuite les coefficients à l'aide des formules suivantes :

$$a = \frac{\displaystyle\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\displaystyle\sum_{i=1}^{n}(x_i - \overline{x})^2} \qquad b = \overline{y} - a\overline{x}$$

Exercice 4 : On reprend les 3 points de nos exemples précédents, {(0,0),(1,1),(2,1)}.

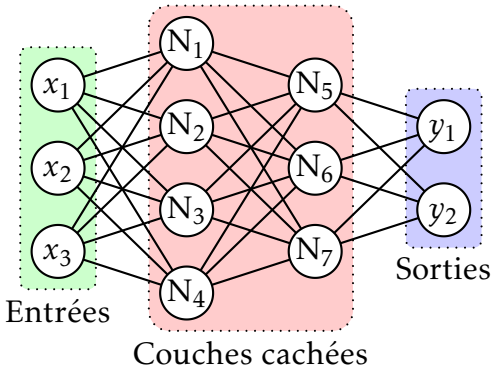
1) Calculer $\overline{x}$ et $\overline{y}$.
2) Compléter le tableau ci-contre.
3) En déduire les valeurs de a et de b.

x_i	y_i	$x_i - \overline{x}$	$y_i - \overline{y}$	$(x_i - \overline{x})^2$	$(x_i - \overline{x})(y_i - \overline{y})$
0	0				
1	1				
2	1				
			Totaux :		

C'est cette méthode qui est utilisée par les calculatrices pour calculer les équations des droites de régression.

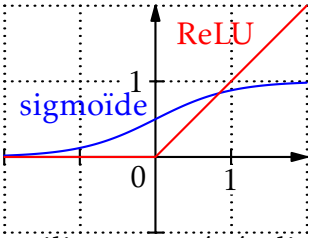
Réseaux de neurones

Un **réseau de neurones** est une fonction qui prend des entrées (3 dans l'exemple)et renvoie des sorties (2 dans l'exemple). Entre les deux, il y a des couches cachées. Chaque couche est composée de neurones. La valeur d'un neurone est calculée à l'aide d'une combinaison linéaire des valeurs de la couche précédente, sur laquelle est appliquée une fonction non linéaire.



Concrétement, la valeur du neurone N_1 est calculée ainsi : $N_1 = f(a_1x_1 + a_2x_2 + a_3x_3 + b_1)$.
Les nombres a_1, a_2 et a_3 sont les poids des liens entre les entrées et N_1. Il y a un poids sur chaque lien entre deux neurones. Sur l'exemple au dessus, il y en a 30. Le nombre b_1 est le biais de N_1, qui permet de le rendre plus ou moins sensible. Chaque neurone, à part les entrées, en a un. Il y en a donc 9 sur l'exemple.

La fonction f est une fonction non linéaire, appelée **fonction d'activation**, qui permet par exemple de limiter les valeurs de N_1 entre $[0;1]$ (avec une sigmoïde), ou de supprimer les valeurs négatives (avec une ReLU). Sans ces fonctions, les couches cachées seraient inutiles et le réseau se ramènerait à une relation linéaire entre les entrées et les sorties.



Afin de trouver des valeurs correctes pour chaque paramètre, on peut utiliser une généralisation de la régression linéaire, appelée **rétropropagation du gradient**. Pour cela, on utilise un jeu de valeurs d'entraînement. Pour chaque entrée, on calcule la somme des carrés des différences entre les résultats obtenus et ceux attendus. On calcule ensuite la dérivée de la fonction de coût obtenue pour chacun des paramètres. On obtient ainsi le **gradient** qui indique comment doit évoluer chaque paramètre. On continue jusqu'à converger vers un minimum local ou au bout d'un certain nombre d'itérations.
On teste alors le réseau obtenu à l'aide d'un autre jeu de tests. Si le résultat est acceptable, on peut utiliser le réseau de neurones, sinon on reprend de nouvelles valeurs aléatoires pour les paramètres et on recommence l'apprentissage, en changeant éventuellement le pas, ou la structure du réseau.
Si l'apprentissage et la mise au point du réseau peut être complexe et longue, l'utilisation d'un réseau de neurones est très efficace puisque reposant principalement sur des fonctions affines. C'est pourquoi des appareils comme des téléphones portables peuvent utiliser sans problème des réseaux de neurones pour reconnaître des visages ou la voix.