De la régression linéaire à l'intelligence artificielle

Rémi Lajugie

mail prenom.nom@ac-limoges.fr

Novembre 2019, version non finalisée

Résumé

Dans cet article nous nous intéressons au modèle de régression linéaire qui permet de trouver la meilleure manière de représenter un nuage de points par une fonction affine. Nous commençons par nous intéresser au cas de la régression linéaire simple en dimension 2 puis nous intéressons à la régression affine.

Nous montrons ensuite comment la régression linéaire peut être employée pour effectuer de la classification binaire supervisée.

Nous nous intéressons ensuite à la généralisation du modèle linéaire au delà de la dimension 2.

Nous pensons que cet article est abordable par un élève de premier cycle universitaire scientifique (le public ciblé étant l'étudiant de «maths sup»). Cet article nous semble également, pour peu que l'on omette les développements mathématiques de la partie 4, pouvoir servir de support à un projet réalisé dans le cadre de la spécialité Numérique et Sciences Informatiques (NSI) de l'enseignement secondaire français. Cet article est indépendant de la précédente note sur les k plus proches voisins, que nous avions rédigée il y a quelques années. Néanmoins sa lecture est conseillée car elle donne un autre éclairage sur des concepts similaires. Il peut également donner des idées exploitables pour un sujet de travail d'initiative personnel encadré (TIPE) en deuxième année de classe préparatoire MP.

Il y a une décennie, la presse parlait d'informatique, d'internet, mais pas d'intelligence artificielle. Aujourd'hui, il ne se passe pas une journée sans que l'on retrouve dans les médias des articles parlant d'intelligence artificielle, de réseaux de neurones, d'algorithme de traitement automatisé des données. L'apparente nouveauté du domaine est renforcé par le fait que la presse utilise les termes anglais : «neural network», «deep learning», «big data», comme si le domaine était tellement neuf qu'il n'y avait pas eu de traductions de termes anglais, issus directement du monde de la recherche. En réalité, l'intelligence artificielle (aussi appelé apprentissage automatique ou apprentissage artificiel ou «machine learning» en anglais) est une discipline beaucoup moins neuve que l'on peut l'imaginer puisqu'on peut sans vraiment exagérer la faire remonter aux travaux de Turing dans les années 1940-1950 (Turing, 1948)
.

Dans cet article, nous allons même faire des connexions avec des méthodes encore plus anciennes (Gauss, 1809 ; Legendre, 1805), puisque nous allons expliquer les liens entre la régression linéaire, bien connue des physiciens, chimistes, biologistes et autres scientifiques manipulant des données expérimentales, et l'apprentissage automatique, c'est à dire ce que l'on appelle l'intelligence artificielle.

Nous allons plus particulièrement nous attacher à la notion de régression linéaire [1] . Cette dernière, bien connue de tous ceux qui font des expériences et doivent trouver la meilleur approximation affine

1. Le terme régression linéaire s'applique indifféremment à la recherche d'une fonction linéaire ou d'une fonction affine. Il s'agit d'un abus de langage renforcé par le fait qu'en anglais, «linear function» désigne ce que nous appelons en français «fonction affine». Dans cet article nous avons fait le choix de séparer les notions de régression affine et régression linéaire.

de leurs résultat, a donné naissance à quelques uns des algorithmes les plus robustes [2] d'apprentissage statistique.

Dans une première partie nous allons détailler le principe de la régression linéaire en dimension 2 en insistant sur sa formulation comme un problème de minimisation du résidu quadratique. Puis nous étendrons cette méthode au cas affine, toujours en dimension 2. Ensuite nous généraliserons à une dimension n quelconque. Enfin nous verrons que la régression linéaire permet de faire de l'apprentissage automatique et nous verrons la nécessité d'introduire la notion de *régularisation*.

1 Introduction

1.1 Principe de la méthode.

Un exemple simple. Considérons le problème suivant : on souhaite établir s'il existe une forme de relation entre la température et l'altitude.

Pour cela on dispose de paires d'observations (altitude/température) comme les points bleus de la figure 1.1.

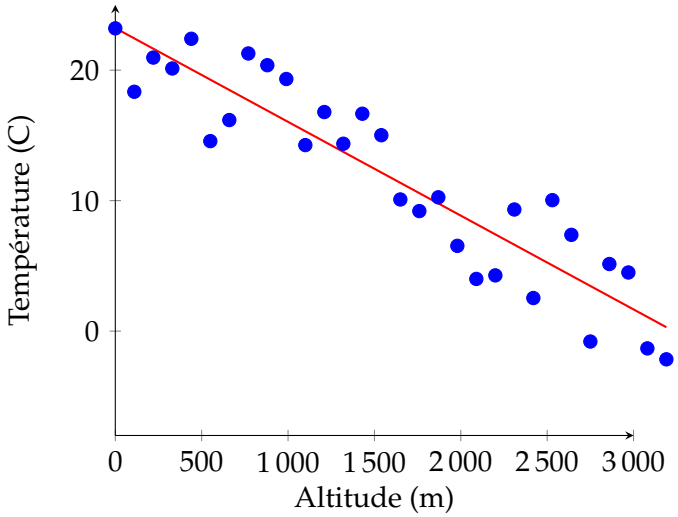


FIGURE 1 – Données fictives (mais réalistes) correspondant à des paires d'observations (altitude/température).

A partir de ces observations, on souhaite déterminer un *modèle* de la relation entre altitude et température : c'est à dire une fonction mathématique f qui prend en entrée une altitude et renvoie une température. Ainsi dans notre exemple : la donnée sera l'altitude et la réponse la température attendue.

La fonction f s'appelle la *fonction de régression*.

Dans la suite, nous utiliserons la lettre x pour faire référence à l'entrée (ici l'altitude) et à y pour faire référence à la réponse, ici la température.

Ajustement d'un modèle affine. Un coup d'oeil à ces données indique qu'un modèle pertinent semble être celui correspondant graphiquement à une droite, autrement dit, une fonction de régression f affine, de la forme $f(x) = ax + b$ et telle que $y \approx f(x) \approx ax + b$. Dans la figure 1.1 on a tracé une droite qui semble une bonne approximation des données observées.

2. Un algorithme est robuste si on peut facilement l'utiliser dans des contextes variés, différents de celui pour lequel il a initialement été conçu.

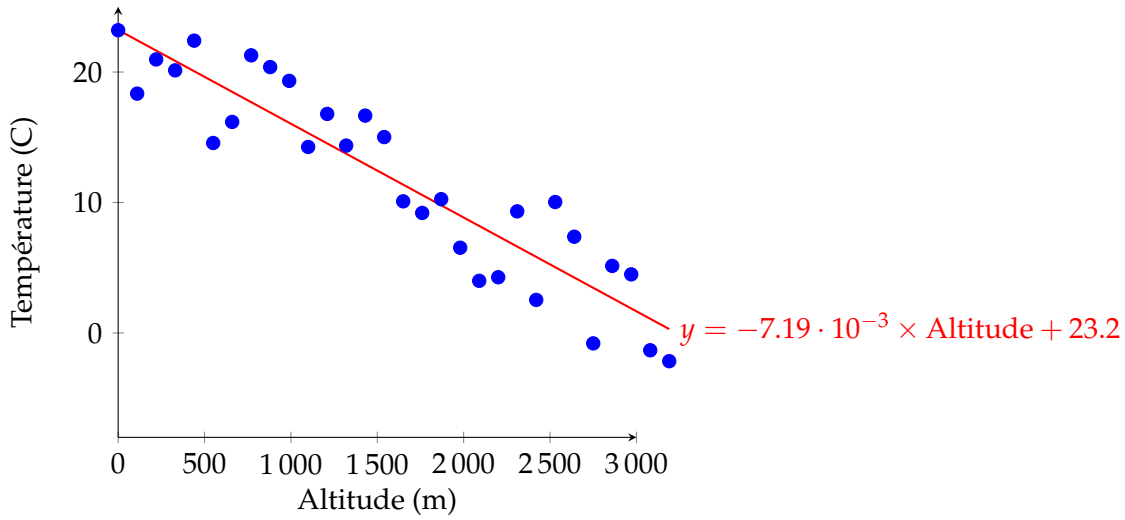


FIGURE 2 – Illustration du principe de la régression linéaire dans un nuage de points en deux dimensions : il s'agit de trouver la droite qui approche le mieux le nuage.

Il reste maintenant à déterminer un critère pour pouvoir sélectionner les coefficients a et b les plus pertinents au regard des observations.

Les résidus quadratiques. Pour cela, on va avoir recours à la méthode *d'estimation par les moindres carrés*.

Supposons que l'on dispose de n paires (x_i, y_i), chacune correspondant à un couple (altitude, température). Nous rappelons que x_i est l'entrée et y_i la sortie, ou réponse, du modèle.

Soit a, b deux paramètres définissant un modèle affine pour les données. On appelle *résidu quadratique* associé aux données, la somme des écarts entre les observations de température y_i et la température $ax_i + b$ prédite par le modèle. Autrement dit, il s'agit de la quantité :

$$E(a, b) = \sum_{i=1}^{n} (y_i - ax_i - b)^2.$$

La méthode d'estimation par les moindres carrés, revient à rechercher les paramètres a, b qui minimisent la quantité E.

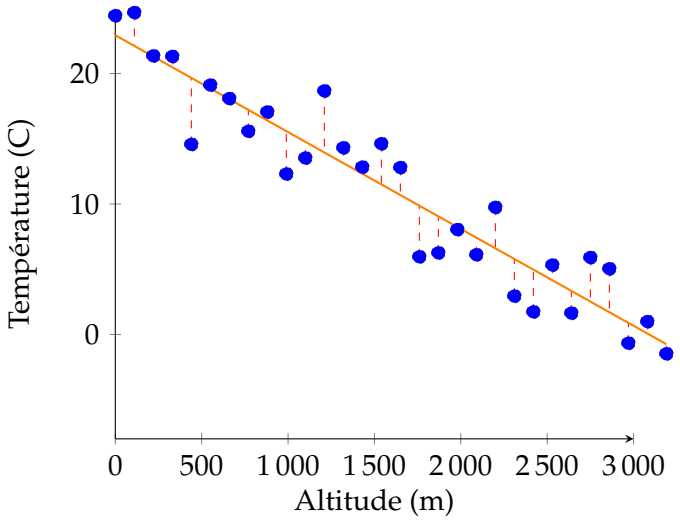


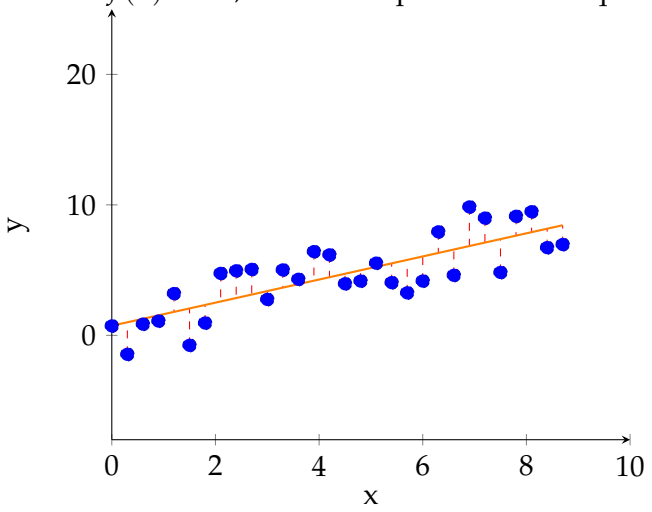
FIGURE 3 – On a représenté ici les résidus. Le but va être de minimiser leur somme.

Dans la suite, nous allons voir comment mathématiquement on peut résoudre ce problème d'optimisation.

2 Régression linéaire en dimension 2

Nous allons commencer par nous placer dans le cas le plus simple où on observe des entrées réelles et où on observe des sorties également réelles (donc on est en dimension 2, comme pour les couples altitude/température).

Dans un premier temps allons chercher, non pas à trouver deux paramètres a et b mais un seul, c'est à dire que nous allons considérer la recherche du meilleur modèle linéaire. La fonction de régression est de la forme $f(x) = ax$, où a est un paramètre réel que l'on cherche à estimer.



2.1 Formulation variationnelle

Formulation du problème d'optimisation. Autrement dit, on cherche à minimiser $E(a) = \sum_{i=1}^{n}(y_i - ax_i)^2$.

On cherche donc à résoudre le problème d'optimisation suivant :

$$\min_{a\in\mathbb{R}} \sum_{i=1}^{n}(y_i - ax_i)^2.$$

Résolution explicite. On sait qu'une condition nécessaire pour que $E(a)$ soit minimale est que sa dérivée $E'(a)$ soit nulle [3]. Donc dérivons.

$$\forall a \in \mathbb{R}, E'(a) = \sum_{i=1}^{n}(-2y_i x_i + 2ax_i),$$

un minimum de cette quantité est atteint pour $E'(a) = 0$. Autrement dit :

$$\begin{aligned} & \sum_{i=1}^{n}(-2y_i x_i + 2ax_i^2) &&= 0 \\ \Leftrightarrow \quad & a\sum_{i=1}^{n} x_i^2 &&= \sum_{i=1}^{n} y_i x_i \quad . \\ \Leftrightarrow \quad & a &&= \frac{\sum_{i=1}^{n} y_i x_i}{\sum_{i=1}^{n} x_i^2} \end{aligned}$$

L'avant-dernière ligne s'appelle *l'équation normale de la régression linéaire.*

Nous avons donc résolu de manière explicite le problème d'optimisation de la régression linéaire, c'est à dire l'ajustement du meilleur modèle linéaire.

3. En réalité cette condition est également suffisante dans notre cas particulier. Le lecteur curieux pourra se convaincre que cette fonction est convexe.

3 Régression affine

Restons en dimension deux avec des paires (x, y) et cherchons désormais à effectuer une régression *affine*.

3.1 Formulation variationnelle

On cherche donc deux coefficients a, b tels que le résidu quadratique sur les données soit minimal.

Résolution explicite. Avant d'aller plus loin, on a besoin de quelques notations. E dépend de deux variables a et b. La dérivée par rapport à la variable a de la fonction obtenue en «gelant»[4] la variable b est appelée la dérivée partielle de E par rapport à a et est notée : $\frac{\partial E}{\partial a}$.

Cette fois, nous devons minimiser suivant deux inconnues a et b. La théorie de la minimisation des « fonctions convexes à deux variables » nous garantit que E sera minimale si $\frac{\partial E}{\partial a} = 0$ et $\frac{\partial E}{\partial b} = 0$.

On calcule ces deux quantités (pour $\frac{\partial E}{\partial a}$ on dérive par rapport à a en traitant b comme si c'était une constante et pour $\frac{\partial E}{\partial b}$ on fait la même chose en traitant a comme une constante).

L'expression des dérivées partielles est la suivante :

$$\begin{aligned}\frac{\partial E}{\partial a} &= \sum_{i=1}^{n} \frac{\partial (y_i^2 + a^2 x_i^2 + b^2 - 2a x_i y_i - 2b y_i + 2a x_i b)}{\partial a} \\ &= \sum_{i=1}^{n} 2a x_i^2 - 2 x_i y_i + 2 x_i b\end{aligned}$$

$$\begin{aligned}\frac{\partial E}{\partial b} &= \sum_{i=1}^{n} \frac{\partial (y_i^2 + a^2 x_i^2 + b^2 - 2a x_i y_i - 2b y_i + 2a x_i b)}{\partial b} \\ &= \sum_{i=1}^{n} 2b - 2 y_i + 2a x_i\end{aligned}$$

En écrivant la condition d'optimalité (à savoir que les dérivées partielles doivent être nulles), on obtient donc deux équations :

$$\sum_{i=1}^{n} a x_i^2 - x_i y_i + x_i b = 0, \tag{1}$$

et (on inverse la relation pour exprimer b directement en fonction de a) :

$$b = \frac{1}{n} \left(\sum_{i=1}^{n} y_i - a \sum_{i=1}^{n} x_i \right). \tag{2}$$

On remplace b dans (1) par l'expression que l'on a trouvé dans l'équation (2).

On obtient alors :

$$0 = \sum_{i=1}^{n} a x_i^2 - \sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} x_i \frac{1}{n} (\sum_{j=1}^{n} y_j - a \sum_{j=1}^{n} x_j).$$

Nous allons effectuer plusieurs opérations permettant d'y voir plus clair dans cette expression :

— la sommation étant linéaire, on peut mettre en facteur a dans les termes de la forme $\sum_{i=1}^{n} a x_i^2$ (qui ne sont jamais une réécriture de $a x_1^2 + a x_2^2 + \ldots + a_n x_n^2$) et donc $\sum_{i=1}^{n} a x_i^2 = a \sum_{i=1}^{n} x_i^2$,

— par commodité on introduit les quantités : $S_{x^2} = \sum_{i=1}^{n} x_i^2$; $S_x = \sum_{i=1}^{n} x_i$; $S_y = \sum_{i=1}^{n} y_i$; $S_{xy} = \sum_{i=1}^{n} x_i y_i$.

4. C'est à dire en considérant comme étant constante.

Au bilan, l'expression devient :

$$0 = aS_{x^2} - S_{xy} + \frac{1}{n}\left(S_x S_y - a(S_x)^2\right).$$

On en déduit une expression de a :

$$a = \frac{S_{xy} - \frac{1}{n}S_x S_y}{S_{x^2} - \frac{1}{n}(S_x)^2}.$$

On trouve ensuite

$$b = \frac{1}{n}\left(S_y - \frac{S_{xy} - \frac{1}{n}S_x S_y}{aS_{x^2} - \frac{1}{n}(S_x)^2} \times S_x\right)$$

Si l'on souhaite implémenter cette méthode en PYTHON, on produirait un code similaire à celui-ci :

```
def regressionAffine(X,Y):
    #Trouve la meilleur fonction affine au sens des moindres carrés
    #X est supposé être le vecteur des entrées et Y celui des réponses
    n = len(X)
    Sy = sum(Y)
    Sx = sum(X)
    Sx2 = sum(X*X)
    Sxy = sum(X*Y)
    a = (Sxy-(1/n)*Sx*Sy)/(Sx2-(1/n)*Sx*Sx)
    b = (1/n)*(Sy-a*Sx)
    return a,b
```

4 En dimension p

Pour le moment, les modèles de régression étudiés ne permettent de trouver un modèle linéaire que pour des données de dimension 1 et des réponses de la même dimension. En règle général, il arrive souvent que l'on ait une seule réponse (par exemple la température atmosphérique) mais que l'on ait plusieurs données pour chaque réponse (l'altitude mais également la pression atmosphérique, le taux d'humidité etc.). Le modèle que nous avons présenté ne s'applique plus tel quel. Le but de cette partie est de montrer comment on peut généraliser le concept de régression linéaire au cas où l'on observe une réponse y_i mais où les données d'entrée sont composées de p descripteurs : $x_1^i, x_2^i, \ldots, x_p^i$ (ces descripteurs sont appelés les « features »en anglais). Par commodité on introduit le vecteur $X_i = \left(x_1^i, x_2^i, \ldots, x_p^i\right)$.

Un modèle linéaire en dimension p sera la donnée de p coefficients $a_1, \ldots, a_p$ tels que la fonction de régression soit $f(X_i) = a_1 x_1^i + \ldots + a_p x_p^i$, soit encore en introduisant le vecteur $a = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{pmatrix}$ par

$$f(X_i) = {}^t a X_i.$$

Laissons de côté pour le moment la question de la régression affine, nous verrons qu'en fait elle peut se réduire à une régression linéaire.

La minimisation des résidus quadratiques conduit donc à minimiser la quantité suivante :

$$E(a) = \sum_{i=1}^{n} \left(y_i - f(X_i)\right)^2 = \sum_{i=1}^{n} \left(y_i - {}^t a X_i\right)^2.$$

Si on introduit le vecteur $y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$ et la matrice $X = \begin{pmatrix} x_1^1 & x_1^2 & \dots & x_1^n \\ x_2^1 & x_2^2 & \dots & x_2^n \\ \vdots & \vdots & \dots & \vdots \\ x_p^1 & x_2^p & \dots & x_p^n \end{pmatrix}$, cette quantité peut s'interpréter comme la norme Euclidienne[5] au carrée de $y - {}^taX$, autrement dit

$$E(a) = \|y - {}^taX\|_2^2 = {}^tyy + {}^taX^tXa - {}^taXy - {}^ty{}^tXa \underbrace{=}_{\text{car } {}^taXy \text{ est un scalaire}} = {}^tyy + {}^taX^tXa - 2{}^taXy$$

Pour trouver le paramètre optimal, comme en dimension 2 il va falloir que toutes les dérivées partielles (le gradient) soient nulles.

L'annexe mathématiques nous permet de calculer le gradient $\nabla E(a)$ en un clin d'oeil :

$$\nabla E(a) = 2X^tXa - 2Xy.$$

Écrivons désormais la condition d'annulation du gradient :

$$\begin{aligned} & \nabla E(a) &=& 0 \\ \Leftrightarrow\ & 2X^tXa - 2{}^tXy &=& 0 \\ \Leftrightarrow\ & X^tXa &=& {}^tXy \\ \Leftrightarrow\ & a &=& \left(X^tX\right)^{-1}{}^tXy \text{sous réserve d'inversibilité de } X^tX \end{aligned}$$

L'avant-dernière équation s'appelle *l'équation normale de la régression linéaire.*

Elle nous assure que le problème de régression linéaire peut se résoudre par la résolution d'un système linéaire de taille $p \times p$ (en théorie inversible avec forte probabilité si $n \geq p$), chose qui peut se faire au moyen de l'algorithme du pivot de Gauss, de complexité $\Theta(p^3)$.

Un code PYTHON permettant d'effectuer une régression linéaire est le suivant :

```
1 def regressionLineaire(X,Y):
2     XT = np.transpose(X)
3     return np.linalg.solve((np.dot(XT, X)), np.dot(XT, Y))
```

4.1 Lien régression affine en dimension p et régression linéaire en dimension $p+1$

Dans cette partie nous considérons une fonction de régression qui est, non pas linéaire, mais affine, autrement dit

$$f(X_i) = {}^taX_i + b = \sum_{k=1}^{p} a_k x_k^i + b,$$

où a est un vecteur de taille p et b un scalaire.

On constate alors qu'en rajoutant un 1 dans le vecteur X_i des descripteurs, c'est à dire si on considère

5. La norme euclidienne d'un vecteur u, de composantes $u_1, \dots, u_p$ en dimension p est la quantité, notée $\|u\|$ égale à $\sqrt{\sum_{i=1}^n u_i^2}$

le vecteur de taille $p+1$: $\tilde{X}_i = \begin{pmatrix} x_1^i \\ x_2^i \\ \vdots \\ x_p^i \\ 1 \end{pmatrix}$, alors l'apprentissage d'un modèle affine en les X_i revient en l'apprentissage d'un modèle *linéaire* sur les $\tilde{X}_i$ (un tel modèle s'écrivant en effet : $f(\tilde{X}_i) = \sum_{k=1}^{p} a_k x_k^i + a_{p+1}1$.

5 Régression linéaire et apprentissage automatique : la régression linéaire pénalisée

Pour le moment, la régression linéaire n'apparaît pas clairement comme un problème lié à l'intelligence artificielle. Dans cette partie, nous faisons plus clairement la connexion.

5.1 Evaluation d'un modèle linéaire

Jusqu'à présent, la régression nous est apparu comme un moyen *commode* pour décrire la relation entre des données d'entrée et des données de sortie (pensons par exemple à des données liant l'altitude à la température), l'ajustement d'un modèle linéaire permet de donner une bonne approximation de données *déjà relevées*.

Pour passer à l'étape supérieure et vraiment pouvoir parler d'apprentissage automatique : il faut que le modèle de régression estimé sur des données passées soit capable de se *généraliser* sur des données futures.

Retour aux questions d'altitude et de température. Reprenons notre exemple en dimension 2 : nous avons effectué une régression linéaire sur des observations passées (des paires altitude/température) et nous nous demandons si désormais le modèle linéaire estimé est une bonne prédiction pour de nouvelles données. Autrement dit, si nous allons à une certaine altitude, la température observée « collera »-t-elle à ce que le modèle linéaire estimé prévoit ?

Dans la figure suivante :

— On retrouve les données sur lesquelles on a ajusté le modèle linéaire,

— on retrouve également de nouvelles observations qui n'ont pas servi à l'ajustement du modèle linéaire.

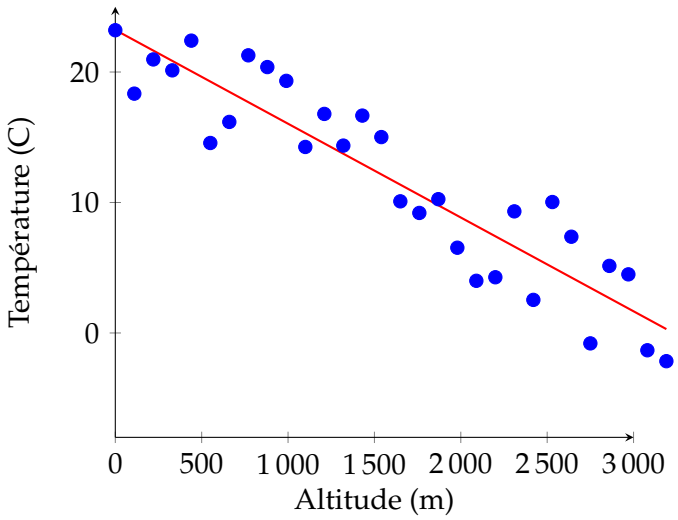


FIGURE 4 – Les points en bleu sont les données dites d'entraînement, correspondant aux observations sur lesquelles on a ajusté le modèle de régression linéaire. Les triangles en rouge sont les données de test, qui sont de nouvelles observations. Le modèle de régression linéaire sera jugé cohérent si ses performances sur les nouvelles données sont de bonne qualité.

De manière plus générale : l'évaluation du modèle. On distinguera désormais deux phases distinctes :

1. la phase d'*entraînement*, durant laquelle on va ajuster un modèle à partir d'observations pour lesquelles on connaît la donnée et la réponse,
2. la phase d'évaluation, durant laquelle on évalue, sur de *nouvelles paires données/réponses*, la qualité du modèle.

Continuons à spécifier le vocabulaire :
— les données et les réponses servant à ajuster le modèle forment l'ensemble d'« entraînement »,
— les données et les réponses servant à évaluer le modèles seront appelées forment l'ensemble de « test ».

Le test de tout algorithme d'apprentissage supervisé (c'est à dire utilisant en entrée des paires données/étiquettes) se fait de la manière suivante : après avoir appris une certaine fonction sur l'ensemble d'entraînement, on demande à cette règle les réponses qu'elle donne sur l'ensemble de test et on compare avec la réponse réelle attendue.

Évaluation du modèle de régression linéaire. Pour un modèle de type régression linéaire, il est naturel de calculer l'erreur quadratique moyenne commise sur l'ensemble de test. C'est à dire, si on note (X_j, y_j) les N paires de données de test et a le vecteur de paramètre :

$$R(a) = \frac{1}{N}\sum_{j=1}^{N}(y_j - {}^t a X_j).$$

5.2 La régression linéaire pénalisée (« ridge regression » en anglais

Le problème du surapprentissage. Pour bien comprendre d'où viennent les problèmes, nous allons ici nous fonder sur un exemple que nous avons simulé en PYTHON [6]

— On fixe ici un vecteur a de dimension 1 (et de composantes $a = (2, 0, 3, -1, 0, 0, 7, 0, 0, 1, 0, 0, 1, -5, 0, 0, 0, 0, 0,$ puis on considère des données générées de la manière suivante : on génère aléatoirement un vecteur x_i et on associe à ce vecteur une réponse $y_i = {}^t a x + \epsilon$ où ϵ est un bruit, c'est à dire une valeur

6. Le code PYTHON ayant produit les diverses figures est disponible sur demande.

aléatoire, tirée au hasard autour de 0. Elle permet de rendre compte des incertitudes qui sont toujours présentes dans des données réalistes. Le vecteur a, qui correspond au modèle sous-jacent, sera dans la suite appelé le « vrai » modèle.

— L'ensemble d'entraînement est composé de 30 données générées comme décrit plus haut, l'ensemble de test de 2000 données.

— Nous ajustons ensuite un modèle de régression linéaire sur les données d'entraînement et relevons les erreurs quadratiques moyennes sur les ensemble d'entraînement et de test.

Voici les performances tronquées à l'unité (la meilleure performance est celle correspondant à la plus petite valeur) du vecteur a estimé par les moindres carrés en comparaison du « vrai ».

	entrainement	test
estimé	81	531
« vrai »	162	234

TABLE 1 – Performances des coefficients estimés et des coefficients optimaux, c'est à dire ceux qui sont vrais. Plus la valeur est petite, meilleure est la performance, estimée comme le résidu quadratique moyen de la régression.

Quelque chose doit nous frapper dans la table précédente : sur le jeu de données d'entraînement, il est possible de trouver des coefficients donnant des résultats meilleurs que les vrais. En revanche, ces coefficients ont de moins bonnes performances sur des données de test. En fait, les paramètres ajustés sur les données d'entraînement collent beaucoup mieux à celles-ci que le vrai paramètre. En revanche elles vont moins bien se généraliser.

Regardons ce qu'il se passe au niveau des coefficients sur cet exemple (les valeurs décimales ont été tronquées à la première décimale significative) :

estimé	1.7	-0.1	3.0	-1.4	-0.5	0.09	6.7	0.1	-0.12	0.5	0.2	0.1	0.6	-4.9	0.5	0.3	0.2	0.1	-0.03	-0.2
vrai	2	0	3	-1	0	0	7	0	0	1	0	0	1	-5	0	0	0	0	0	0

Si certains coefficients sont correctement estimés, d'autres sont fantaisistes.

Intuitivement, il faut s'imaginer que 30 points en dimension 20 remplissent beaucoup moins l'espace que 30 points en dimension 2 ; il y a donc plus de « place » entre les points d'apprentissage et fatalement le modèle qui colle le plus aux données n'est pas forcément le meilleur au sens que l'on souhaiterait.

Il s'agit de ce que l'on appelle la « malédiction de la dimensionnalité » ou le « surapprentissage » : le paramètre estimé par la régression linéaire est trop «compliqué». Comme on l'a dit précédemment, les coefficients ajustés sur les données d'entraînement collent trop aux données. On dit qu'ils ont du mal à généraliser [7].

Dans cet exemple, de nombreux coefficients sont égaux à 0, ce qui traduit le fait que certains paramètres n'ont pas d'incidence sur la prédiction [8]

Cela suggère de sélectionner a tel que a ait beaucoup de coefficients nuls et n'ait pas des coefficients de valeur trop grande.

L'idée clef : la pénalisation du modèle de régression linéaire. C'est là que rentre en scène le concept de *pénalisation*. Pour empêcher d'avoir des coefficients « trop grands », conduisant à une règle de prédiction « trop compliquée », on va *pénaliser* le modèle de la régression linéaire par une quantité liée au

7. En général, on compare cela à l'apprentissage par coeur chez les humains : à apprendre la solution d'un exercice sans la comprendre, on aura de très bonne performances si on refait le même exercice. En revanche, on aura beaucoup plus de mal si on nous présente une nouvelle situation, légèrement différente.

8. C'est une situation extrêmement courante dans la vie réelle, où l'on dispose pour chaque observation d'une unique réponse associée à des données de grande dimension. Prenons l'exemple d'une entreprise de vente en ligne qui souhaite connaître l'état du compte en banque d'un de ses clients. Elle connaît peut être son adresse, son nom, son prénom, son âge, sa profession, sa taille, son poids, ses achats précédents, sa consommation de thé, son rythme cardiaque etc. Toutes ces données ne sont pas forcément pertinentes pour la prédiction du solde bancaire.

poids des coefficients. En pratique, on utilise la pénalité suivante :

$$\lambda \|a\|_2^2 = \lambda \sum_{i=1}^{n} a_i^2,$$

où λ est un réel positif.

Dans l'annexe mathématique, nous justifions, avec des arguments accessibles en première année de classe préparatoire, l'emploi de cette pénalité $\lambda\|a\|^2$.

Remarquons que plus les coefficients de a sont grands, plus cette quantité (que l'on appelle la *norme euclidienne* de a) sera grande.

La fonction à minimiser est désormais celle-ci :

$$E_\lambda(a) = \sum_{i=1}^{n} (y_i - {}^t aX)^2 + \lambda \|a^2\|_2^2 = \|y - {}^t aX\|^2 + \lambda \|a^2\|_2^2.$$

Pour l'instant, on se garde bien de dire comment choisir λ (il doit quand même être positif). Remarquons que le choix de cette pénalité a plusieurs propriétés et intérêts :

— Un intérêt mathématique : E_λ est une fonction convexe et pour la minimiser, il suffit de mettre son gradient à 0.
— Quand $\lambda = 0$, on retrouve la régression linéaire.
— Quand λ tend vers $+\infty$, $\sum_{i=1}^{n}(y_i - {}^t aX)^2$ devient négligeable devant $\lambda\|a^2\|_2^2$ et E_λ se comporte comme l'application $a \mapsto \|a\|_2^2$ dont le minimum est le vecteur nul 0.

Un calcul similaire à celui fait pour la régression non pénalisée nous conduit à la où I_p désigne la matrice identité de taille p :

$$({}^t XX + \lambda I_p)a = {}^t Xy.$$

Les mathématiques nous garantissent que cette dernière équation a, pour λ non nul toujours une et une seule solution. En effet, la matrice $(X^t X)$ a toujours des valeurs propres supérieures ou égales à 0 donc $(X^t X + \lambda I_p)$ a toutes ses valeurs propres supérieures ou égales à λ, donc en particulier 0 n'est pas valeur propre et la matrice est inversible.

Un code PYTHON permettant d'effectuer une régression linéaire pénalisée est le suivant :

```
1 def regressionRidge(X,Y, lam):
2     #lam correspond au paramètre lambda de régularisation du modèle ridge
3     XT = np.transpose(X)
4     return np.linalg.solve((np.dot(XT, X)+lam*np.eye(XT.shape[0])), np.dot(XT, Y)
```

Chemin de régularisation. Sur un exemple en dimension 4 sur des données synthétiques, où les coefficients de régression optimaux de régression sont $(0,1,2,3)$, on a représenté, en fonction de λ, la valeur estimée des coefficients de la régression linéaire pénalisée : c'est ce que l'on appelle le *chemin de régularisation*.

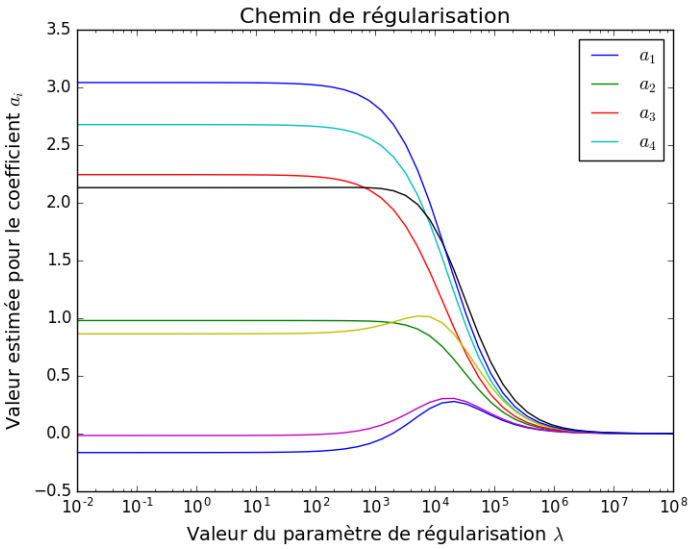


FIGURE 5 – Asymptotiquement quand λ tend vers l'infini, le vecteur de régression tend vers 0.

Courbes d'apprentissage. Dans ce paragraphe, nous considérons en dimension 50 un ensemble d'apprentissage de points et un ensemble de test de 2000 points. Nous représentons ici l'évolution du résidu quadratique moyen sur l'ensemble d'entraînement et de test (la performance est d'autant meilleure que le résidu moyen est faible).

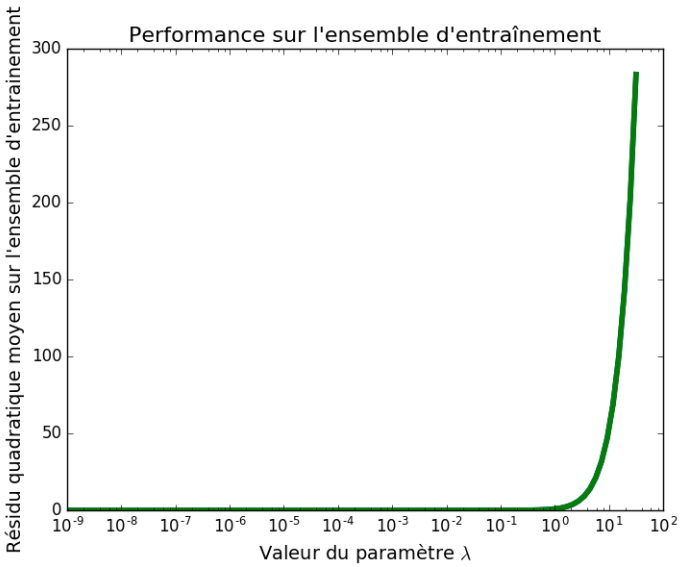


FIGURE 6 – Evolution du résidu quadratique moyen sur l'ensemble d'entraînement.

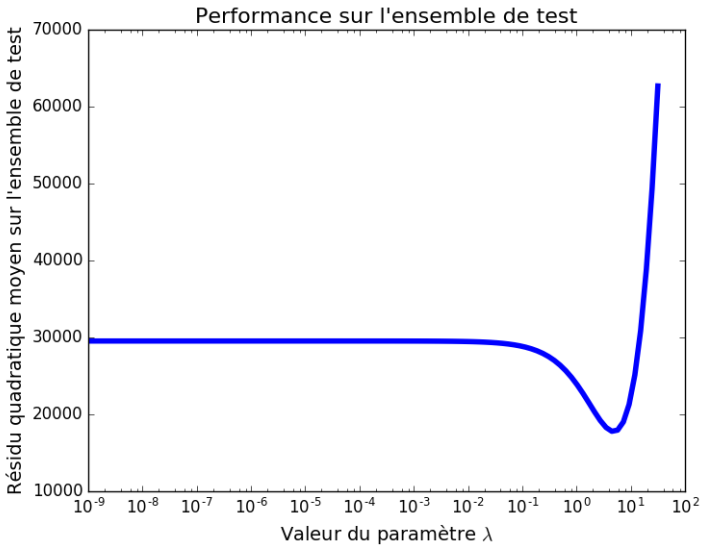


FIGURE 7 – On constate que la meilleure performance est attente pour λ non nul. Quand on va vers la gauche, on observe le phénomène de surapprentissage : on colle trop aux données ; quand on va vers la droite, c'est a contrario le phénomène de sous-apprentissage : le vecteur se rapproche du vecteur nul et le modèle devient proche d'un modèle constant.

Les courbes d'apprentissage ont ici des formes caractéristiques :

— Sur les données d'entrainement l'erreur croît avec la valeur du paramètre estimé λ, ce qui est tout à fait attendu puisque plus λ est faible plus le problème d'optimisation résolu ressemble à une régression non pénalisée.

— Sur les données de test, on observe qu'il y a un *compromis* à trouver, des valeurs de λ trop petite conduisent à trop coller aux données, c'est le *surapprentissage* tandis que des valeurs trop grandes conduisent à une mauvaise performance, c'est le sous-apprentissage.

Conclusion

En guise de conclusion, nous pouvons dire qu'au travers de l'exemple de la régression linéaire pénalisée, nous avons observé le concept central de l'apprentissage machine moderne : celui de surapprentissage.

Tous les algorithmes d'intelligence artificielles, aussi sophistiqués qu'ils soient, sont en lutte avec ce phénomène.

Dans l'exemple de la régression linéaire, l'ajout d'un paramètre de régularisation permet, sous réserve de l'ajuster correctement, d'éviter l'écueil du surapprentissage.

Néanmoins nous n'avons pas dit comment il était possible d'ajuster le paramètre λ de la régression pénalisée. Il suffit d'avoir recours à la méthode de *validation* simple. On sépare l'échantillon de données d'entraînement en données d'entraînement et données de validation. Pour différentes valeurs de λ, on ajuste le paramètre de régression sur les données d'entraînement et on évalue ses performances sur les données de validation.

On verra dans une prochaine note, comment faire mieux que la validation simple, qui a le défaut de *réduire* la taille des données disponibles.

Le lecteur intéressé pourra se plonger avec profit dans l'ouvrage suivant : (**?**).

6 Annexe mathématique

6.1 Calcul différentiel en dimension n

Dans cette partie, nous nous intéressons au gradient de certaines fonctions de plusieurs variables. On rappelle que, pour une f application de $\mathbb{R}^n \to \mathbb{R}$, le gradient est le vecteur des dérivées partielles :

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}.$$

Dérivation d'une forme linéaire en dimension n. Dans ce paragraphe, nous nous plaçons dans l'espace vectorielle $\mathbb{R}^n$, nous fixons un vecteur $a = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \dots \\ a_n \end{pmatrix}$

On considère l'application suivante :

$$\begin{array}{rccc} f & : & \mathbb{R}^n & \to & \mathbb{R} \\ & & x & \mapsto & {}^t a x \end{array}.$$

On remarque que pour tout $x \in \mathbb{R}^n$, $f(x) = \sum_{i=1}^n a_i x_i$.

Prenons $k \in [\![1, n]\!]$. Calculons la dérivée partielle d'une telle fonction par rapport à la variable x_k. On obtient :

$$\frac{\partial f}{\partial x_k} = a_k.$$

Le vecteur gradient est donc tout simplement $\nabla f(x) = a$. Il est constant sur l'espace $\mathbb{R}^n$ tout entier.

Gradient d'une fonction quadratique symétrique vectorielle. Dans ce paragraphe nous considérons une matrice $S \in \mathcal{M}_n(\mathbb{R})$ qui est symétrique (c'est à dire que pour tous $i, j \in [\![1, n]\!]$ on a $S_{i,j} = S_{j,i}$.).

On considère l'application suivante :

$$\begin{array}{rccc} q & : & R^n & \to & \mathbb{R} \\ & & x & \mapsto & {}^t x {}^t S x \end{array}.$$

Un produit matriciel donne que pour tout $x \in \mathbb{R}^n$, $q(x) = \sum_{j=1}^n \sum_{i=1}^n x_i x_j S_{i,j}$.

Prenons $k \in [\![1, n]\!]$. Calculons la dérivée partielle d'une telle fonction par rapport à la variable x_k. On obtient :

$$\frac{\partial q}{\partial x_k} = \sum_{i=1}^n x_i S_{i,k} + \sum_{i=1}^n x_i S_{k,i} = 2 \sum x_i S_{i,k},$$

où la dernière inégalité provient de la symétrie de S.

Or on reconnaît que $\sum x_i S_{i,k}$ est le coefficient d'indice i de ${}^t x S$ (ou de Sx par symétrie de S Le vecteur gradient est donc $\nabla q(x) = 2Sx$.

6.2 Justification du choix de la pénalisation dans le modèle linéaire

Reprenons la fonction que l'on doit minimiser pour effectuer une régression linéaire pénalisée.

$$E_\lambda(a) = \sum_{i=1}^{n}(y_i - {}^t aX)^2 + \lambda\|a^2\|_2^2 = \|y - {}^t aX\|^2 + \lambda\|a^2\|_2^2.$$

La pénalité par une quantité proportionnelle à la norme de $\|a\|^2$ peut se faire de la manière suivante :

— une fonction de régression $f : x \mapsto {}^t ax$ est considérée comme «complexe» si ses variations sont grandes,

— or si l'on regarde pour $x, x' \in \mathbb{R}^n$, la différence entre $f(x)$ et $f(x')$, l'inégalité de Cauchy-Schwarz donne :

$$|f(x) - f(x')| = |{}^t ax - {}^t ax'| = |{}^t a(x - x')| =\leq \|a\|\|x - x'\|,$$

on l'on a noté $\langle u, v\rangle$ le produit scalaire entre deux vecteurs $u, v \in \mathbb{R}^n$.
Cette dernière équation nous dit qu'un a de petite norme donnera une fonction de régression aux variations faibles. De plus, comme l'inégalité de Cauchy-Schwarz est une inégalité lorsque les vecteurs impliqués sont colinéaires, on sait que, pour des vecteurs x, x' tels que $x' - x$ est colinéaire à a, $|{}^t a(x - x')| = \|a\|\|x - x'\|$. Cela nous garantit qu'une fonction de régression ne peut avoir de faibles variations partout dans l'espace $\mathbb{R}^n$ que si et seulement si $\|a\|$ est faible.

— En d'autres termes, et c'est un principe très général, si a a une norme euclidienne *petite*, alors $f : x \mapsto {}^t ax$ aura des variations faibles.

Références

Gauss, C. (1809). *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*. sumtibus Frid. Perthes et IH Besser.

Legendre, A.-M. (1805). Appendice sur la méthodes des moindres carrés. *Nouvelles méthodes pour la détermination de l'orbite des comètes*, pages 72–80.

Turing, A. M. (1948). Intelligent machinery.