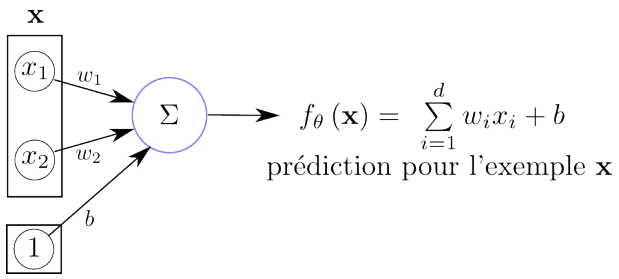
Mod Lin - TP n°1

Dans ce premier TP, nous étudierons quelques variantes de l'algorithme de la régression linéaire / polynomiale.

Exercice n°1 : Régression linéaire et intervalle de confiance

La régression linéaire est une des plus vieilles méthodes statistiques connues. Encore aujourd'hui, elle reste extrêmement populaire du fait d'un bon compromis entre efficacité et facilité de mise en œuvre.

Le point de départ de cette méthode est un problème de régression pour lequel on dispose d'un jeu de données $\mathcal{D} = \left\{\left(\mathbf{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n}$. Les exemples $\mathbf{x}^{(i)}$ appartiennent à espace $\mathbb{X} = \mathbb{R}^d$. Les réponses $y^{(i)}$ appartiennent à un espace $\mathbb{Y} = \mathbb{R}$. Notre but est de trouver une fonction linéaire $f_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}$:



Pour des commodités de notation, on suppose avoir **rajouté un 1** à la fin de chaque vecteur $\mathbf{x}$ qui dans cette hypothèse est à présent de taille $d+1$. En outre, nous noterons

$$\boldsymbol{\theta} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \\ b \end{pmatrix}. \tag{1}$$

Maintenant que nous avons un modèle, nous allons voir comment obtenir une solution satisfaisante, c'est à dire trouver une bonne valeur de $\boldsymbol{\theta}$. C'est là qu'interviennent nos données $\mathcal{D}$:

- Intuitivement, on veut $\hat{y}^{(i)} = f_{\boldsymbol{\theta}}\left(\mathbf{x}^{(i)}\right) \approx y^{(i)}$ pour tout i.
- Une façon mathématique de dire ça est de vouloir que $\sum_{i=1}^{n}\left(f_{\boldsymbol{\theta}}\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2$ soit petit.
- Une façon mathématique d'obtenir ça est de poser $J(\boldsymbol{\theta}) = \sum_{i=1}^{n}\left(f_{\boldsymbol{\theta}}\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2$ et de minimiser la fonction J.

La fonction J est appelée **fonction de coût** (concept ultra-important en ML). Le succès de la régression linéaire vient du fait qu'il est très facile de trouver le minimum de J. On montre en effet sans difficulté que le minimiseur $\boldsymbol{\theta}^*$ de J est l'unique point tel que :

$$\frac{dJ}{d\boldsymbol{\theta}}(\boldsymbol{\theta}^*) = 0. \tag{2}$$

Grâce à des résultats d'algèbre linéaire, résoudre cette équation est tout aussi facile. Pour commencer, observez que

$$f_{\boldsymbol{\theta}}\left(\mathbf{x}^{(i)}\right) = \boldsymbol{\theta}^T \cdot \mathbf{x}^{(i)} \text{ (produit scalaire)}. \tag{3}$$

Adoptons également, une notation vue en cours :

matrice des exemples $\mathbf{X}=$

$$\begin{array}{c|c|c|c|}
\mathbf{x}^{(1)} & \mathbf{x}^{(2)} & & \mathbf{x}^{(n)} \\
\hline
x_1^{(1)} & x_1^{(2)} & & x_1^{(n)} \\
x_2^{(1)} & x_2^{(2)} & & x_2^{(n)} \\
x_3^{(1)} & x_3^{(2)} & \cdots & x_3^{(n)} \\
\cdots & \cdots & & \cdots \\
x_d^{(1)} & x_d^{(2)} & & x_d^{(n)}
\end{array}$$

dimension d

nbr d'exemple n

De la même manière, notons

$$\mathbf{y} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{pmatrix}. \tag{4}$$

On peut alors écrire

$$J(\boldsymbol{\theta}) = \left(\mathbf{X}^T\boldsymbol{\theta} - \mathbf{y}\right)^T \cdot \left(\mathbf{X}^T\boldsymbol{\theta} - \mathbf{y}\right) \tag{5}$$

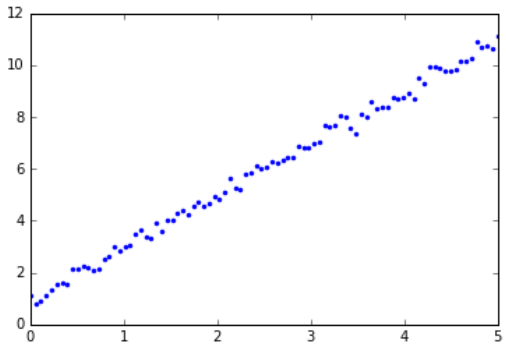
1. A l'aide du formulaire de dérivation matricielle, montrez que :

$$\frac{dJ}{d\boldsymbol{\theta}}(\boldsymbol{\theta}) = 2\mathbf{X}\left(\mathbf{X}^T\boldsymbol{\theta} - \mathbf{y}\right). \tag{6}$$

   En déduire que

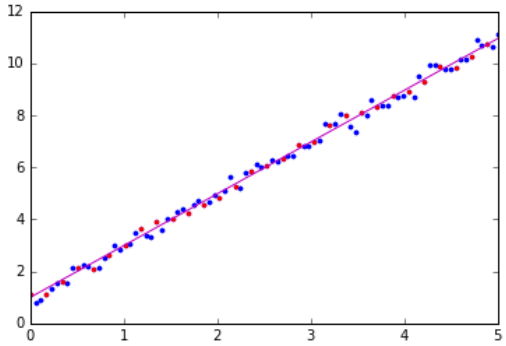
$$\boldsymbol{\theta}^* = \left(\mathbf{X}\mathbf{X}^T\right)^{-1}\mathbf{X}\mathbf{y}. \tag{7}$$

2. Nous allons à présent tester l'efficacité de cette méthode sur un jeu de donnée synthétique. Répartissez n =90 points sur l'intervalle $[0;5]$ (fonction `numpy.linspace`). Ces points constituent nos exemples et pour bien visualiser le comportement cet algorithme, nous travaillerons avec $d = 1$
3. Générez les réponses $y^{(i)}$ selon une distribution Gaussienne : $Y|X = x^{(i)} \sim \mathcal{N}\left(2x^{(i)} + 1, \sigma\right)$ avec $\sigma = 0.2$. Affichez ces données. Voici un exemple de visualisation souhaitée :

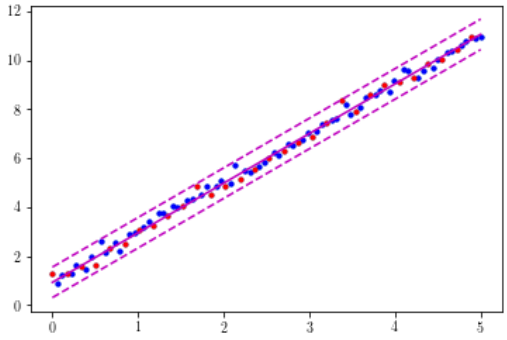


4. Découpez à présent ce dataset en train et en test avec un ratio $\frac{1}{3}$ / $\frac{2}{3}$. On conservera les noms de variable habituels :
   — `X_train` de type `numpy array` 1D contenant les exemples d'apprentissage,
   — `X_test` de type `numpy array` 1D contenant les exemples de test,
   — `y_train` de type `numpy array` 1D contenant les réponses des exemples d'apprentissage,
   — `y_test` de type `numpy array` 1D contenant les réponses des exemples de test.
   Le découpage se fera de manière déterministe (boucle `for`, tous les 3 tours x_i est attribué à `X_train` et à `X_test` le reste du temps).

5. Calculez θ^* selon la formule de la question 1. Attention, veillez à augmenter la variable `X_train` avec une ligne de 1 (fonctions `numpy.ones` et `numpy.vstack`). Les opérations d'algèbre linéraire se feront aussi via le module `numpy` : `numpy.dot` et `numpy.linalg.inv`
6. Affichez la droite estimée d'équation $\theta_1^* x + \theta_2^*$. Voici un exemple de visualisation souhaitée :



7. On enrichit à présent le modèle d'une vision probabiliste en faisant l'hypothèse que $Y|X \sim \mathcal{N}(\hat{y}, \sigma)$. Calculer l'estimé du maximum de vraisemblance du paramètre σ (voir formule (27) du cours). L'estimé est il proche de la valeur théorique utilisé dans le vrai modèle génératif des données ?
8. Dans un modèle gaussien, l'intervalle de demi-longueur 3σ autour de l'espérance contient 99% des réalisations de la variable gaussienne. Tracer les bornes inférieures et supérieures de cet intervalle autour de notre droite :



Exercice n°2 : Régression polynomiale et régularisation Ridge

Nous allons à présent généraliser l'algorithme de la régression linéaire au cas **polynomial**, c'est à dire qu'on suppose que la relation liant les $\mathbf{x}^{(i)}$ aux y_i est un polynôme de degré p supérieur à 1. Pour simplifier la présentation supposons que $p = 2$ et surtout que $d = 1$, c'est à dire que

$$f_{\boldsymbol{\theta}}(\mathbf{x}) \quad = \quad w_{1,2}x^2 + w_{1,1}x + b \tag{8}$$

L'indexation du paramètre $w_{1,k}$ fait référence au terme du polynôme de degré k pour l'unique dimension de $\mathbf{x}$. Dès qu'on choisit $d > 1$, le nombre de termes du polynôme croît très rapidement. Par exemple, pour $d = 2$, on aurait :

$$f_{\boldsymbol{\theta}}(\mathbf{x}) \quad = \quad w_{2,2}x_1^2x_2^2 + w_{2,1}x_1^2x_2 + w_{1,2}x_1x_2^2 + w_{2,0}x_1^2 + w_{0,2}x_2^2 + w_{1,1}x_1x_2 + w_{1,0}x_1 + w_{0,1}x_2 + b. \tag{9}$$

Dans tous les cas, on remarque qu'on peut ré-écrire l'expression de la fonction $f_{\boldsymbol{\theta}}(\mathbf{x})$ dans une forme proche de celle de la régression linéaire. Pour le cas de l'équation (8), posons

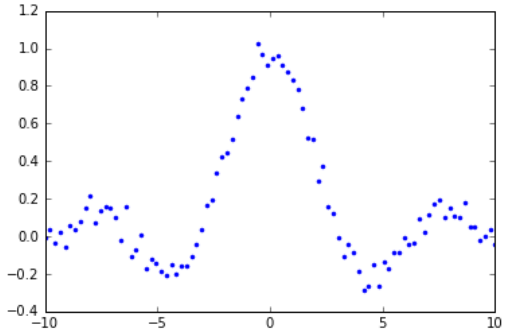
$$\boldsymbol{\theta} = \begin{pmatrix} w_{1,2} \\ w_{1,1} \\ b \end{pmatrix} \text{ et } \mathbf{z} = \begin{pmatrix} x^2 \\ x \\ 1 \end{pmatrix}. \tag{10}$$

On a donc

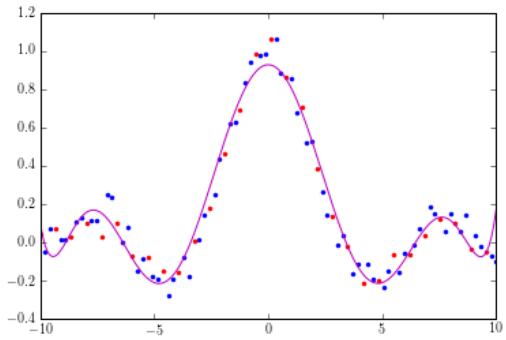
$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T\mathbf{z}. \tag{11}$$

En voyant $f_{\boldsymbol{\theta}}$ comme une fonction de $\mathbf{z}$, on se ramène bien au cas linéaire. Les vecteurs $\mathbf{z} \in \mathcal{Z}$ forment une représentation intermédiaire et $\mathcal{Z}$ est appelé **espace d'attributs**.

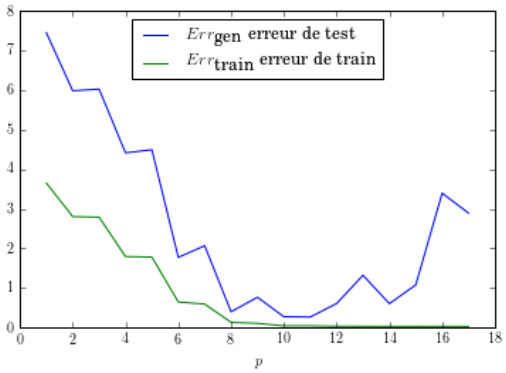
1. Nous allons essayer d'appliquer une régression polynomiale à la fonction $\frac{\sin(x)}{x}$. Comme précédemment, nous allons commencer par générer des données. Répartissez n =90 points sur l'intervalle $[-10; 10]$.
2. Générez les réponses y_i selon une distribution Gaussienne : $Y|X = x \sim \mathcal{N}\left(\frac{\sin(x)}{x}, \sigma\right)$ avec $\sigma = 0.05$. Affichez ces données. Voici un exemple de visualisation souhaitée :



3. Créez des ensembles de train et de test de la même manière que pour le cas linéaire.
4. Créez une fonction `polyreg` qui prend en entrée `X_train`, `y_train` et le degré du polynôme p. Cette fonction retourne le vecteur $\boldsymbol{\theta}^*$. Pour le calculer, il suffit d'appliquer l'équation (7) en remplaçant les $\mathbf{x}^{(i)}$ par les $\mathbf{z}^{(i)}$.
5. Utilisez la fonction `polyreg` sur vos données avec $p = 8$. Tracez le polynôme correspondant au vecteur $\boldsymbol{\theta}$ retourné par la fonction. Voici un exemple de visualisation souhaitée :

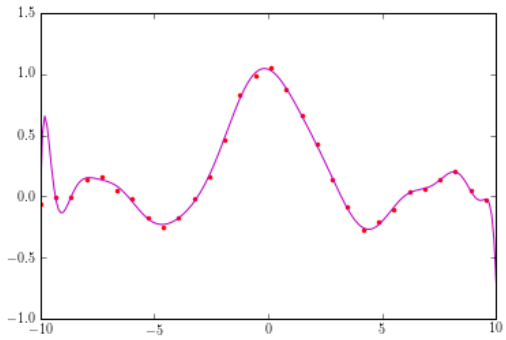


6. Calculez et affichez les erreurs de train et de test en faisant varier p de 1 à 17. Vous devez obtenir des courbes semblables aux suivantes :



D'après ce graphique, quel est le degré optimal pour cette regression polynomiale ?

7. Tracez le polynôme correspondant au degré 19 en le superposant aux données d'apprentissage seulement. Voici un exemple de visualisation souhaitée :



En quoi ce résultat illustre le phénomène de sur-apprentissage (overfitting) ?

8. Si on adopte l'approche bayésienne avec un prior gaussien, la solution du problème s'écrit alors

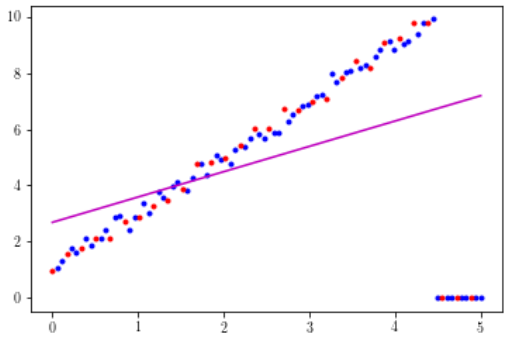
$$\boldsymbol{\theta}^* = \left(\mathbf{X}\mathbf{X}^T + \lambda \mathbf{Id}\right)^{-1} \mathbf{X}\mathbf{y}, \tag{12}$$

où l'hyperparamètre $\lambda = \frac{1}{2\sigma_0}$ dans les notations du cours. Mettez en place cette solution alternative, reposant sur une régularisation dite Ridge. Déterminez une valeur de λ qui atténue l'overfitting dans la situation obtenue en question précédente.

Exercice n°3 : Régression linéaire robuste et RANSAC

Il arrive relativement souvent qu'un dataset soit pollué par la présence de données aberrantes souvent issues d'erreurs de mesure ou d'un défaut matériel. Dans cet exercice, on se propose d'entraîner une régression linéaire dans de telles conditions sans avoir à nettoyer à la main le dataset.

1. Reprenez le dataset de l'exercice 1 et remplacez les 10 dernières valeur du vecteur de label par zéro. Le dataset ressemble dorénavant à



2. Vérifiez qu'en entraînant une régression linéaire (sans régularisation), on obtient une droite similaire à celle en magenta dans la figure ci dessus.
3. Pour rétablir un apprentissage de qualité, nous allons utiliser l'algorithme RANSAC. Dans le cadre de cet algorithme,
   — vous commencerez à entraîner une régression linéaire avec seulement les trois premiers points de l'ensemble d'apprentissage.
   — une fois la première estimation de θ^* obtenue, vous déterminerez quels exemples d'apprentissage appartiennent à l'intervalle de demi-longueur $3\hat{\sigma}_{mle}$
   — sur la base des exemples d'apprentissage retenus à l'étape précédente, vous ré-estimez θ^* et vous revenez à l'étape précédente.

   En déterminant le critère d'arrêt de votre choix, programmez RANSAC et montrer qu'on rétablit la situation.