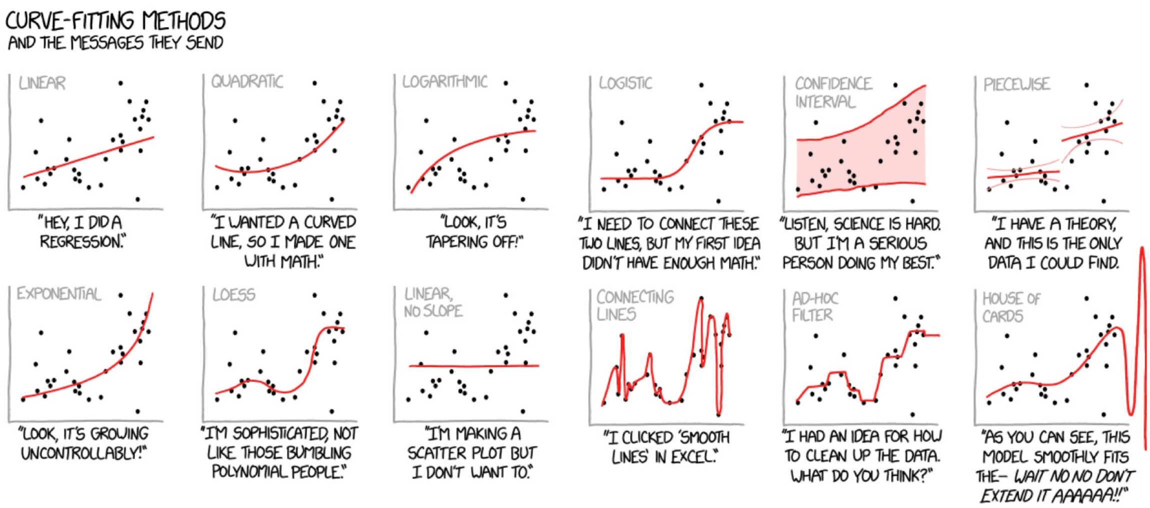
PROJET 4

Régressions

sujet proposé par Tony Lelièvre

tony.lelievre@enpc.fr



La partie simulation comptera pour la moitié de la note.

L'objectif de ce projet est d'explorer différents aspects des méthodes de régression : régression linéaire (première partie), puis régression logistique (deuxième partie). Dans la deuxième partie, on réalisera l'apprentissage par un algorithme de gradient stochastique. Les deux parties sont indépendantes l'une de l'autre.

1 Régression linéaire

On dispose d'observations $(x_i, y_i)_{1\leq i\leq n} \in \mathbb{R}^{2n}$ que l'on souhaite représenter par un modèle linéaire : pour tout $1 \leq i \leq n$,

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i \tag{4.1}$$

avec $(\epsilon_i)_{i\geq 1}$ une suite i.i.d. de variables aléatoires gaussiennes centrées de variance $\sigma^2 > 0$ supposée connue. L'objectif est d'estimer $\beta = (\beta_0, \beta_1) \in \mathbb{R}^2$ à partir des observations $(x_i, y_i)_{1\leq i\leq n}$.

S1. Écrire une fonction prenant en paramètres $\beta \in \mathbb{R}^2$, n et σ et générant des observations $(x_i, y_i)_{1\leq i\leq n}$ suivant le modèle (4.1). On tirera les x_i de manière i.i.d selon la loi uniforme sur $(0, 1)$. Représenter graphiquement le résultat sous la forme d'un nuage de points $(x_i, y_i)_{1\leq i\leq n}$ pour $\sigma = 0.1$, $n = 100$ et $\beta = (2, 1)$.

T1. On suppose que les $(x_i)_{1\leq i\leq n}$ sont fixés. Vérifier que $y = (y_i)_{1\leq i\leq n}$ est un vecteur aléatoire de densité

$$p(y) = (2\pi\sigma^2)^{-n/2} \exp\left(-\frac{\|y - X\beta\|^2}{2\sigma^2}\right)$$

où $X \in \mathbb{R}^{n\times 2}$ est la matrice définie par : $X_{i,1} = 1$ et $X_{i,2} = x_i$ pour tout $i \in \{1, \dots, n\}$. Ici, et dans toute la suite, les vecteurs sont des vecteurs colonnes et $\|\cdot\|$ désigne la norme euclidienne. L'estimateur du maximum de vraisemblance du paramètre β est donc donné par

$$\hat{\beta} \in \arg\min_{\beta\in\mathbb{R}^2} \|y - X\beta\|^2. \tag{4.2}$$

T2. Montrer que si $\hat{\beta}$ satisfait (4.2), alors

$$X^T X\hat{\beta} = X^T y.$$

Vérifier que $\mathrm{Im}(X^T X) = \mathrm{Im}(X^T)$ et en déduire l'ensemble des solutions de (4.2).
Indication : on pourra vérifier et utiliser le fait que la fonction $\beta \to \|y - X\beta\|^2$ est convexe et différentiable sur $\mathbb{R}^2$, et donc l'ensemble des minima globaux de cette fonction coïncide avec l'ensemble des points qui annulent le gradient de cette fonction.

On suppose dans toute la suite que la matrice $X^T X$ est inversible (on le vérifiera numériquement si besoin). On a donc

$$\hat{\beta} = (X^T X)^{-1} X^T y.$$

T3. Vérifier que $y = X\beta + \epsilon$ où $\epsilon = (\epsilon_i)_{1\leq i\leq n}$. Pour X et β fixés, quelle est la loi de $\hat{\beta}$? En déduire que $\mathbb{E}(\hat{\beta}) = \beta$.

T4. Montrer que $\sigma^{-2}(\hat{\beta} - \beta)^T X^T X(\hat{\beta} - \beta)$ suit une loi du Chi² à 2 degrés de liberté (notée $\chi^2(2)$). Soit $q_{\chi^2(2)}(r)$ le quantile d'ordre $r \in [0, 1]$ de la loi $\chi^2(2)$. Pour $\alpha \in [0, 1]$, on introduit l'ellipsoïde de confiance à l'ordre α :

$$\mathcal{E}_\alpha = \left\{\tilde{\beta} \in \mathbb{R}^2, \sigma^{-2}(\tilde{\beta} - \hat{\beta})^T (X^T X)(\tilde{\beta} - \hat{\beta}) \leq q_{\chi^2(2)}(1 - \alpha)\right\}.$$

Prouver que $\mathbb{P}(\beta \in \mathcal{E}_\alpha) = 1 - \alpha$.

S2. Implémenter un algorithme pour représenter l'ellipsoïde de confiance pour $\alpha = 0.05$, $\sigma = 0.1$, $n = 100$ et $\beta \in \{(0, 1), (1, 0), (2, 1)\}$, en utilisant les observations générées à la première question. Vérifier que le niveau de confiance avec lequel l'ellipsoïde a été construit est effectivement observé numériquement. Comment évoluent vos résultats numériques en fonction du nombre d'observations n ?

On s'intéresse désormais au comportement des estimateurs dans la limite $n \to \infty$, et on indique donc explicitement la dépendance en n de X, y, $\hat{\beta}$ et ϵ : X^n, y^n, $\hat{\beta}^n$ et ϵ^n.

T5. En utilisant la loi forte des grands nombres, montrer que presque sûrement

$$\lim_{n\to\infty} \frac{1}{n}(X^n)^T X^n = \begin{bmatrix} 1 & 1/2 \\ 1/2 & 1/3 \end{bmatrix} \text{ et } \lim_{n\to\infty} \frac{1}{n}(X^n)^T \epsilon^n = 0.$$

En déduire que presque sûrement

$$\lim_{n\to\infty} \hat{\beta}^n = \beta.$$

S3. Illustrer numériquement la convergence de $\hat{\beta}^n$ vers β.

2 Régression non-linéaire

On s'intéresse désormais à des observations $(x_i, y_i)_{1\leq i\leq n} \in \mathbb{R}^{2n}$ que l'on souhaite représenter par un modèle non-linéaire (régression logistique) : pour tout $1 \leq i \leq n$,

$$y_i = \frac{1}{1+\exp(-\theta_0 - \theta_1 x_i)} + \epsilon_i \tag{4.3}$$

avec $(\epsilon_i)_{i\geq 1}$ une suite i.i.d. de variables aléatoires gaussiennes centrées de variance $\sigma^2 > 0$ supposée connue. L'objectif est d'estimer $\theta = (\theta_0, \theta_1) \in \mathbb{R}^2$ à partir des observations $(x_i, y_i)_{1\leq i\leq n}$.

S4. Ecrire une fonction prenant en paramètres $\theta \in \mathbb{R}^2$, n et σ et générant des observations $(x_i, y_i)_{1\leq i\leq n}$ suivant le modèle (4.3). On tirera les x_i de manière i.i.d selon la loi uniforme sur $(-3, 3)$. Représenter graphiquement le résultat sous la forme d'un nuage de points $(x_i, y_i)_{1\leq i\leq n}$ pour $\sigma = 0.1$, $n = 200$ et $\theta = (0.8, 1.7)$.

On considère désormais que les données $(x_i, y_i)_{1\leq i\leq n}$ sont fixées, et on cherche à estimer θ.

T6. Vérifier que l'estimateur du maximum de vraisemblance du paramètre θ est donné par

$$\hat{\theta} \in \arg\min_{\theta\in\mathbb{R}^2} L(\theta)$$

où la fonction de perte (*loss function*) est

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n} |y_i - f(\theta, x_i)|^2 \text{ avec } f(\theta, x) = \frac{1}{1+\exp(-\theta_0 - \theta_1 x)}.$$

S5. Programmer une fonction qui prend en paramètres θ et $(x_j, y_j)_{1\leq j\leq n}$, et qui rend $L(\theta)$. Représenter graphiquement la fonction de perte $\theta \in \mathbb{R}^2 \mapsto L(\theta)$ en utilisant les points générés à la question **S4**.

En pratique, si le nombre de données est trop important (n grand), on ne considère pas toutes les données mais seulement un sous-ensemble $I_m \subset \{1, \ldots, n\}$ de cardinal $m \leq n$, et on introduit le gradient *mini-batch* :

$$g_m(\theta) = \nabla_\theta \left(\frac{1}{m} \sum_{j\in I_m} |y_j - f(\theta, x_j)|^2 \right).$$

Noter que

$$g_m(\theta) = \frac{1}{m}\sum_{j\in I_m} G_j \text{ avec } G_j = 2(f(\theta, x_j) - y_j)\nabla_\theta f(\theta, x_j). \tag{4.4}$$

Par ailleurs, on notera dans la suite $g(\theta) = \nabla_\theta L(\theta) = \frac{1}{n}\sum_{i=1}^{n} G_i$ le gradient de la fonction de perte.

S6. Programmer une fonction qui prend en paramètres θ, $(x_j, y_j)_{1\leq j\leq n}$ et m et qui rend $g_m(\theta)$, les indices I_m étant choisis aléatoirement suivant une des deux méthodes suivantes :

- *Tirage avec remplacement*: I_m est constitué de m indices tirés de manière indépendante et uniforme dans $\{1, \ldots, n\}$;
- *Tirage sans remplacement*: I_m est constitué de m indices tirés de manière uniforme dans $\{1, \ldots, n\}$ sans remplacement (on ne peut pas tirer deux fois le même indice).

On pourra utiliser la commande `numpy.random.choice`. Vérifier que la fonction rend le bon vecteur en la testant en un point θ quelconque, avec $m = n$ et sans remplacement (de sorte que $g_n = g$), en comparant le résultat à des différences finies calculées sur la fonction de perte L (cf. la fonction programmée dans la question précédente).

T7. Dans cette question et la suivante, on s'intéresse à la variable aléatoire $g_m(\theta)$, l'aléa portant uniquement sur le tirage des indices I_m. Montrer que $\mathbb{E}(g_m(\theta)) = g(\theta)$ pour les deux modes de tirage (avec ou sans remplacement).

T8. On introduit la matrice de taille 2×2 :

$$\Sigma(\theta) = \frac{1}{n} \sum_{i=1}^{n} \overline{G}_i \overline{G}_i^T \text{ avec } \overline{G}_i = G_i - g(\theta) \in \mathbb{R}^2.$$

Montrer que $\mathrm{Cov}(g_m(\theta)) = \frac{1}{m}\Sigma(\theta)$ dans le cas avec remplacement, où $\mathrm{Cov}(g_m(\theta))$ désigne la matrice de covariance de $g_m(\theta)$. Montrer que $\mathrm{Cov}(g_m(\theta)) = \frac{1}{m}\frac{n-m}{n-1}\Sigma(\theta)$ dans le cas sans remplacement. Quel estimateur est préférable en terme de variance?

S7. Tracer un histogramme des valeurs de $g_m(\theta)$ sur plusieurs tirages indépendants de I_m, avec ou sans remplacement. Illustrer numériquement les résultats des deux questions précédentes (moyenne et covariance, en fonction de la taille m du *mini-batch*).

S8. On chercher à calculer numériquement $\hat{\theta}$ par l'algorithme de gradient stochastique suivant : pour θ^0 donné, pour $1 \leq k \leq K$, on calcule

$$\theta^k = \theta^{k-1} - \eta\, g_m^k(\theta^{k-1})$$

où $\eta > 0$ est le pas (appelé également *learning rate* dans le contexte du *machine learning*), et les fonctions $(g_m^k)_{k\geq 1}$ sont associées à des tirages i.i.d. d'indices I_m^k parmi $\{1, \dots, n\}$, cf. l'équation (4.4). Programmer cette méthode et représenter graphiquement l'évolution de θ^k par rapport à la valeur cible $\theta = (0.8, 1.7)$ utilisée pour générer les données. Représenter également l'évolution de la fonction de perte $L(\theta^k)$ et de $\arg\min_{(\theta^l)_{1\leq l\leq k}} L(\theta^l)$ en fonction de k. On prendra une condition initiale θ^0 aléatoire, un *mini-batch* de taille $m = 10$ (tirage sans remplacement), un nombre d'itérations $K = 10\,000$ et un pas $\eta = 0.1$.