TELECOM ParisTech **MDI720**
2015/2016 **Joseph Salmon**

Devoir maison n° 1 : Intervalles de confiance et méthodes gloutonnes

---

Pour ce travail vous devez déposer un **UNIQUE** fichier sous format `ipynb` sous EOLE. Vous devez charger votre fichier sur Éole (MDI720 > Validation), avant le 25/10/2015 23h59. La note totale est sur **20** points répartis comme suit :

— qualité des réponses aux questions : **15** pts,

— qualité de rédaction, de présentation et d'orthographe : **2** pts,

— indentation, Style PEP8, commentaires adaptés : **2** pts,

— absence de bug : **1** pt.

Retard : malus de **4** pts par tranche de 24h (sauf excuses validées par l'administration).

Rappel : aucun travail par mail accepté !

Exercice 1. (Intervalle de confiance dans le modèle gaussien)

On considère le jeu de donnée `airquality`, et l'on souhaite expliquer la concentration en ozone en fonction des autres variables disponibles (et de la variable constante). On utilise un modèle linéaire, en supposant les bruits *i.i.d.*, gaussiens, de loi : $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}_n, \sigma^2 \operatorname{Id})$, avec σ inconnue.

1) Écrire mathématiquement le modèle linéaire correspondant.

2) Récupérer le jeu de données à partir du package `statsmodels.datasets`. On pourra utiliser la fonction `get_rdataset` de ce package avec la méthode `data`.

3) Enlever les lignes qui contiennent des valeurs manquantes.

4) Ajuster le modèle par la méthode des moindres carrés (avec `sklearn` ou `statsmodels`), en régressant la variable `'Ozonne'` sur les cinq autres variables. On se placera dans le cas où l'on a centré et réduit les variables explicatives au préalable.

5) Calculer l'estimateur des moindres carrés des coefficients du modèle (ainsi que de l'ordonnée à l'origine) ? Donner l'expression théorique d'un estimateur sans biais de la variance du bruit, puis le résultat numérique obtenu.

On se propose de trouver un intervalle de confiance à 99% de chacun des coefficients pris séparément. On pourra se servir dans la suite de la proposition suivante :

Proposition 1. *On se place dans le modèle linéaire* $Y = X\boldsymbol{\theta}^* + \boldsymbol{\varepsilon}$, *avec* $Y \in \mathbb{R}^n, X \in \mathbb{R}^{n\times(p+1)}$ *et* $\boldsymbol{\varepsilon} \in \mathbb{R}^n$. *On suppose les bruits Gaussiens i.i.d.* $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}_n, \sigma^2 \operatorname{Id})$. *Soient* $\hat{\boldsymbol{\theta}}$ *et* $\hat{\sigma}^2$ *les estimateurs sans biais de* $\boldsymbol{\theta}^*$ *et de* σ^2 *des moindres carrés ordinaires.*

— $\hat{\boldsymbol{\theta}}$ *et* $\hat{\sigma}^2$ *sont indépendants.*

— *Si* $X^\top X$ *est inversible, alors pour tout vecteur* $u \in \mathbb{R}^{p+1}$,

$$\frac{u^\top(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*)}{\hat{\sigma}\sqrt{u^\top(X^\top X)^{-1}u}} \tag{1}$$

suit une loi de Student à $n-p-1$ *degrés de liberté.*

6) Proposer des intervalles de confiance pour chacun des coefficients $\boldsymbol{\theta}_j^*$. Calculer les numériquement pour toutes les variables explicatives. On pourra par exemple utiliser `scipy.stats.t`.

7) Que pouvez-vous en déduire sur la pertinence des variables `'Day'` et `'Month'` (considérées séparément) ?

8) On enregistre une nouvelle observation (Solar.R=197, Wind=10, Temp=70, Day=1, Month=3). Quelle est la prévision du modèle concernant la concentration en ozone ?

Exercice 2. (Intervalle de confiance et *bootstrap*)

On considère encore le jeu de donnée `airquality` et l'on souhaite expliquer la concentration en ozone en fonction des autres variables disponibles (et de la variable constante), par un modèle linéaire, en supposant les bruits centrés, indépendants, de même loi une variance σ^2 inconnue. Ici, on ne suppose pas le bruit gaussien. On va proposer des intervalles de confiance dans un tel contexte, en se basant sur le *bootstrap*. Cette technique fonctionne de la manière suivante : on choisit un paramètre entier B (*e.g.,* $B = 10000$), et l'on crée B échantillons de taille n (n est le nombre d'observations ou nombre de lignes) que l'on obtient en tirant, avec remise, n lignes parmi les lignes initiales. Partant de cette collection d'échantillons, on peut alors utiliser les statistiques empiriques sur celle-ci pour estimer les grandeurs théoriques (quantiles, moyenne, etc.). Plus de détails peuvent être trouvés sur le *bootstrap* dans [Hastie et al., 2009, 7.11, Bootstrap Methods] et là : `https://speakerdeck.com/jakevdp/statistics-for-hackers`.

1) Calculer les estimateurs des coefficients obtenus par une moyenne, puis par une médiane sur les échantillons *bootstrap*. Comparer les avec ceux obtenus avec la méthode de régression classique. On utilisera notamment la fonction `utils.resample` de `sklearn`.

2) Calculer un intervalle de confiance de niveau 99%, en utilisant les quantiles empiriques de niveau 99.5% et 0.5% des échantillons *bootstrap*.

3) Afficher sur un graphe les deux courbes (en pointillés noirs) donnant les bornes inférieures et supérieures (des intervalles de confiance) pour la variable `'Wind'`, et ce en fonction du paramètre B. On ajoutera en trait plein la courbe des estimés par la médiane *bootstrap* de ce même coefficient. On fera varier B de 1 à 5001 par saut de 500.

4) Afficher sur un seul graphique les points et les droites de régression correspondant à faire une régression de la variable `'Ozone'` sur `'Wind'` mois par mois. Quel mois semble atypique ?

   Indication : on pourra utiliser par exemple la fonction `lmplot` de `seaborn`.

Exercice 3. (Algorithmes gloutons ou *greedy*) On se place dans le cadre du modèle linéaire et on centre et reduit X.

1) Écrivez une fonction `stpforward` qui prend comme argument : les observations Y, la matrice X, et enfin un paramètre M qui est le nombre maximum de variables sélectionnées. En sortie, la fonction doit renvoyer la listes S des indices des variables sélectionnées et le vecteur $\boldsymbol{\theta} \in \mathbb{R}^p$ (dont les coordonnées sont nulles sauf les θ_j pour $j \in S$) correspondant à l'estimation des moindres carrés effectuée seulement sur les variables dont les indices sont dans S.

   On codera cet algorithme comme résumé par l'Algorithme 1 : on ajouter itérativement à l'ensemble des variables actives la variable dont le produit scalaire avec les résidus est le plus grand en valeur absolue.

   On utilise la notation X_S pour la matrice créee en extrayant seulement les colonnes de X dont les indices sont dans S. Idem pour $\boldsymbol{\theta}_S$: c'est un vecteur de taille $|S|$, ayant les même valeurs que $\boldsymbol{\theta}$ aux coordonnées indexées par S.

2) Créer une classe `MYOMP` qui implémente `stpforward`, en partant de l'exemple proposé ci-dessous.

Algorithm 1 Algorithme *forward stage-wise selection* (ou *Orthogonal Matching Mursuit*)

Input : observations $\mathbf{y}$, matrice $X = [\mathbf{x}_1, \cdots, \mathbf{x}_p]$; nombre variables sélectionnées : M
Initialiser $\boldsymbol{\theta} = 0$; $r = \mathbf{y}$ et $S = \varnothing$
for $i = 1, \dots, M$ **do**
  Calculer pour tout $j \in [\![1, p]\!] \cap S^c, \alpha_j = |\langle \mathbf{x}_j, r \rangle|$
  Trouver $j_{\max} = \underset{j \in [\![1,p]\!] \cap S^c}{\arg\max}\ \alpha_j$
  $S \leftarrow S \cup \{j_{\max}\}$ (rajout de $i_{\max}$ aux indices retenus)
  $\boldsymbol{\theta}_{\text{int}} \leftarrow \underset{\theta \in \mathbb{R}^{|S|}}{\arg\min} \|\mathbf{y} - X_S \theta\|$ (optionnellement on rajoute les constantes et on récupère θ_0)
  $r \leftarrow \mathbf{y} - X_S \boldsymbol{\theta}_{\text{int}}$
end for
$\boldsymbol{\theta}_S = \boldsymbol{\theta}_{\text{int}}$
Output : $\boldsymbol{\theta} \in \mathbb{R}^p$, $S \subset [\![1, p]\!]$, (optionnellement on peut renvoyer θ_0 l'intercepte).

```python
import numpy as np
from sklearn.linear_model.base import LinearModel, _pre_fit
from sklearn.base import RegressorMixin


class MYOMP(LinearModel, RegressorMixin):

    def __init__(self, n_nonzero_coefs=None, fit_intercept=True,
                 normalize=True, precompute='auto'):
        self.n_nonzero_coefs = n_nonzero_coefs
        self.fit_intercept = fit_intercept
        self.normalize = normalize
        self.precompute = precompute

    def fit(self, X, y):
        """Fit the model using X, y as training data.
        Parameters
        ----------
        X : array-like, shape (n_samples, n_features)
            Training data.
        y : array-like, shape (n_samples,) or (n_samples, n_targets)
            Target values.
        Returns
        -------
        self : object
            returns an instance of self.
        """
        X, y, X_mean, y_mean, X_std, Gram, Xy = \
            _pre_fit(X, y, None, self.precompute, self.normalize,
                     self.fit_intercept, copy=True)
        self.coef_ = np.zeros([X.shape[1], ]) # MODIFY HERE !!!
        self._set_intercept(X_mean, y_mean, X_std)
        return self
```

3) Appliquer `MYOMP` au jeu de données `airquality`, pour $M = 3, 4, 5$.
4) Comparer votre sortie avec celle de `OrthogonalMatchingPursuit` de `sklearn`.
5) Question bonus : utiliser une validation croisée pour choisir le nombre de variables à garder M. Faut-il en garder 1, 2, 3, 4 ou bien 5 ?

Références

[Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning*. Springer Series in Statistics. Springer, New York, second edition. http://www-stat.stanford.edu/~tibs/ElemStatLearn/. 2