université PARIS DIDEROT PARIS 7

REGRESSION LOGISTIQUE
ET
ANALYSE DISCRIMINANTE LINEAIRE

PROGRAMMATION EN PYTHON
MASTER 2 ISIFAR
2019-2020

Christian RAMIRES ALIAGA

I. INTRODUCTION

Dans ce projet, j'abord deux méthodes de classification supervisée (la régression logistique et l'analyse discriminante linéaire) dans le cas binaire. Pour chacune de ces deux techniques je présente, d'une part, les notions théoriques permettant de les comprendre, ainsi que les parties mathématiques que j'ai codé en python. D'autre part, j'utilise mon code sur un jeux de données connu —*Iris de Fisher*— dans le but de constater le bon fonctionnement de mon programme.

En ce qui concerne mon code, j'ai défini uniquement des fonctions pour construire la régression logistique, tandis que je créais une classe et des méthodes pour l'analyse discriminante linéaire. Ainsi, les deux techniques ont été construites de manière différente afin de manipuler plusieurs objets distincts.

II. REGRESSION LOGISTIQUE

a) PRINCIPE DE LA REGRESSION LOGISTIQUE

La régression logistique binaire est une méthode supervisée semi paramétrique. Il s'agit d'une technique qui vise à classer un individu dans un groupe parmi ces deux groupes $\{0, 1\}$.

Soient $\theta^T = (\theta_0, \dots, \theta_p)$ le vecteur des paramètres à estimer, $x^T = (x_1, \dots, x_p)$ le vecteur des variables explicatives et $\phi^T = (1, x_1, \dots, x_p) = (1, x^T)$ où $x_i \in R^n$.

Ce procédé est basé sur le classifieur de Bayes qui consiste à comparer les probabilités a posteriori, en d'autres termes, le point x est affecté à la classe de plus forte probabilité a posteriori.

$$D(x) = argmax_{y \in Y}\, P(y \mid x) \text{ où } Y = \{0,1\}$$

Dans le cas binaire, il faut uniquement la connaissance du rapport des probabilités a posteriori, d'où la règle de décision suivante :

$$D(x) = \begin{cases} 1 & si\ \frac{P(y=0|x)}{P(y=1|x)} > 1 \\ 0 & sinon \end{cases}$$

Le modèle logistique est le plus souvent exprimé par l'expression suivante :

$$\log \frac{P(y=1\,|x)}{P(y=0\,|x)} = \theta_0 + \theta_1 x_1 + \cdots + \theta_p x_p$$

$$= [1\ x^T]\theta = \phi^T \theta$$

Ainsi, la probabilité conditionnelle de chaque valeur de y (0 ou 1), étant donné le vecteur x des variables explicatives, est modélisée suivant la formule :

$$P(y = 1|x) = \frac{e^{\phi^T \theta}}{1 + e^{\phi^T \theta}} = \frac{1}{1 + e^{-\phi^T \theta}}$$

$$P(y = 0|x) = 1 - P(y = 1|x) = \frac{1}{1 + e^{\phi^T \theta}}$$

b) FONCTION SIGMOÏDE

Cette fonction est définie par $\sigma(x) = \frac{1}{1+e^{-x}}$ ou $\sigma(x) = \frac{e^x}{1+e^x}$ pour tout réel x. Elle est importante car je m'en sers pour programmer la régression logistique. Dans mon code, j'ai écrit cette fonction de la manière suivante :

$$\sigma(x) = \begin{cases} \frac{1}{1 + e^{-x}} & si \ x > 0 \\ \frac{e^x}{1 + e^x} & sinon \end{cases}$$

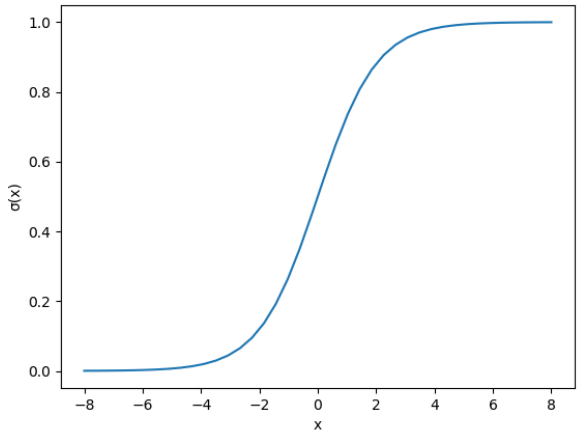

Figure 1 : Fonction sigmoïde

D'après ce qu'on vient d'observer, comme $P(Y = 1|x) = \frac{1}{1+ e^{-\phi^T \theta}}$ alors $P(Y = 1|x) = \sigma(\phi^T \theta)$.

c) ESTIMATION DES PARAMETRES

La méthode du maximum de vraisemblance fournit des valeurs des paramètres qui rendent maximum la probabilité d'obtenir l'ensemble des données observées {(y_i, x_i), i=1, …, n}.

$$L(x_1, \dots, x_p) = \prod_{i=1}^{N} P(y_i, x_i)$$

$$L(\theta;\ x_1, \dots, x_p)\ = \prod_{i=1}^{N} P(y_i|x_i;\ \theta) * p_X(x_i)$$

D'où le logarithme de la vraisemblance :

$$\mathcal{L}(\theta;\ x_1, \dots, x_p)\ = \sum_{i=1}^{N} \log\left(P(y_i|x_i;\ \theta)\right) + \sum_{i=1}^{N} \log\left(p_X(x_i)\right)$$

Le terme de droite ne dépend pas de θ, alors pour déterminer l'estimateur il suffit de maximiser J :

$$\mathrm{J} = \sum_{i=1}^{N} \log\left(P(y_i|x_i;\ \theta)\right)$$

En développant, on obtient l'expression suivante :

$$\mathrm{J} = \sum_{i=1}^{N} y_i \log\big(P(y=1|x_i;\ \theta)\big) + (1-y_i)\log\left(1 - P(y=1|x_i;\ \theta)\right)$$

Pour simplifier les notations, posons $p_i = P(y=1|x_i;\ \theta)$. D'où :

$$\mathrm{J} = \sum_{i=1}^{N} y_i \log(p_i) + (1-y_i)\log\left(1-p_i\right)$$

Étant donné que $P(y_i = 1|x) = \sigma(\phi_i^T\theta)$, alors :

$$\mathrm{J} = \sum_{i=1}^{N} y_i \log(\sigma(\phi_i^T\theta)) + (1-y_i)\log\left(1 - \sigma(\phi_i^T\theta)\right)$$

Dans mon code, au premier abord, j'ai utilisé cette expression pour représenter le logarithme de la vraisemblance sous le nom de ***log_vraisemblance_v1***. Mais la machine a eu des difficultés pour traiter des petits et grands nombres (par exemple : e^{800} ou $\log\left(10^{-400}\right)$. J'ai donc développé l'expression jusqu'à ce qu'on obtienne une plus adéquate.

Ici, $\sigma(\phi_i^T\theta) = \frac{e^{\phi_i^T\theta}}{1+e^{\phi_i^T\theta}}$ alors :

$$\mathrm{J} = \sum_{i=1}^{N} y_i\phi_i^T\theta - \log\left(1 + e^{\phi_i^T\theta}\right)$$

Afin de ne pas générer des erreurs, j'ai modifié légèrement le critère J dans mon code comme vous pouvez le constater ci-dessous. Cette nouvelle version est sous le nom de ***log_vraisemblance_v2*** *:*

$$y_i\phi_i^T\theta - \log\left(1 + e^{\phi_i^T\theta}\right) = \begin{cases} y_i\phi_i^T\theta - \phi_i^T\theta & si\ \ \phi_i^T\theta > 500 \\ y_i\phi_i^T\theta - \log\left(1 + e^{\phi_i^T\theta}\right) & sinon \end{cases}$$

Le log de la vraisemblance est concave, donc on peut lui trouver un maximum. Il n'existe pas de formule analytique pour trouver cette valeur, mais on peut appliquer des méthodes numériques d'approximation, par exemple, les méthodes du gradient, la méthode de Newton-Raphson et l'algorithme BFJS.

d) METHODE DE NEWTON

Cette technique itérative sert à trouver la racine d'une fonction. Dans la régression logistique elle est utilisée afin d'approximer l'estimateur θ en se basant sur la formule de Taylor à l'ordre 2 :

$$J(\theta) \approx J(\theta^k) + (\theta - \theta^k)^T * g_k + \frac{1}{2}(\theta - \theta^k)^T * H_k * (\theta - \theta^k)$$

Où g_k est le gradient et H_k la matrice hessienne de J au point θ^k.
En dérivant J, on obtient :

$$g = \nabla_\theta J(\theta) = \sum_{i=1}^{N} \phi_i(y_i - p_i) \quad o\grave{u} \quad p_i = \sigma(\phi_i^T \theta)$$

$$= \phi^T(y - p)$$

Ainsi que :

$$H = \frac{\partial^2 J(\theta)}{\partial\theta\, \partial\theta^T} = \sum_{i=1}^{N} \phi_i^T * p_i(1 - p_i) * \phi_i$$

$$= -\phi^T W \phi \quad o\grave{u} \quad W = diag(p_i(1 - p_i))$$

Les expressions de g et H sous forme matricielle sont définies dans mon code sous les noms de ***gradient*** et ***hessienne***. Puis, elles sont utilisées dans l'algorithme de Newton pour déterminer θ — ***calcul_estimateur_newton*** — est son nom dans mon script.

Algorithme :

Initialiser $\theta^0, k = 0, nmax = 100\ et\ \epsilon = 10^{-6}$
Tant que $k < nmax$ et $\| \theta^{k+1} - \theta^k \| > \epsilon$ et $\| g_{k+1} - g_k \| > \epsilon$ faire :
    Calculer le gradient $g_k = \nabla_\theta J(\theta^k)$
    Calculer la matrice hessienne $H_k = \frac{\partial^2 J(\theta^k)}{\partial\theta\, \partial\theta^T}$
    $\theta^{k+1} = \theta^k - H_k^{-1} * g_k$
    $k = k + 1$
Fin

e) METHODE DU GRADIENT A PAS FIXE

Un autre recours plus simple pour estimer θ est la méthode du gradient à pas fixe. Elle est élémentaire car elle a uniquement besoin de la connaissance du gradient $\nabla_\theta J(\theta)$ pour faire son travail. Cependant, elle est fortement déconseillée puisqu'elle nécessite d'un grand nombre d'itérations. Dans mon code, j'ai également écrit une fonction pour représenter cet algorithme sous le nom de ***calcul_estimateur_pas_fixe***.

Algorithme :

Initialiser $\theta^0, k = 0, nmax = 1000, \epsilon = 10^{-6}$, ρ fixé
Tant que $k < nmax$ et $\| g_{k+1} - g_k \| > \epsilon$ faire :
    Calculer le gradient $g_k = \nabla_\theta J(\theta^k)$
    $\theta^{k+1} = \theta^k + \rho * g_k$
    $k = k + 1$
Fin

f) REGLE DE DECISION

Dans mon code, la classification d'un nouvel individu « x » repose sur la fonction sigmoïde σ et elle est nommée ***predict_reg*** :

$$D(x) = \begin{cases} 1 & si\ \sigma\left(x^T\hat{\theta}\right) \geq 0.5 \\ 0 & sinon \end{cases}$$

g) APPLICATION

Dans cette section, j'analyse la base de données historique —*Iris de Fisher*— qui est composé de 150 fleurs dont les caractéristiques sont les suivantes : longueur des sépales, largeur des sépales, longueur des pétales, largeur des pétales et l'espèce (setosa, virginica et versicolor). Pour des raisons pratiques – pour tracer des graphiques – j'ai sélectionné les deux premières composantes du dataset et la dernière représentant la variable qualitative des espèces pour former une nouvelle base de données sur laquelle j'ai travaillé. J'ai aussi ré-étiqueté les fleurs de type setosa avec le label 0 et les autres types avec le label 1 car j'explore la régression logistique dans le cas binaire.

Ensuite, j'ai séparé les données en deux groupes : un échantillon d'apprentissage (voir Figure 3) contenant 120 fleurs et un échantillon test (voir Figure 4) composé de 30 fleurs. Puis, à l'aide de mon code et en particulier de ***calcul_estimateur_newton***, j'ai estimé le paramètre $\theta^T = (\theta_0, \theta_1, \theta_2)$ et tracé la frontière de décision (l'hyperplan $\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$) sur l'échantillon d'apprentissage (voir Figure 5).

En ce qui concerne la prédiction des étiquettes de 30 fleurs, j'ai eu comme résultat la matrice de confusion ci-dessous grâce à mon code, ce qui montre qu'il a bien fonctionné (cette performance est envisageable puisque les classes sont linéairement séparables).

	Valeurs réelles	
Valeurs prédites	1	0
1	10	0
0	0	20

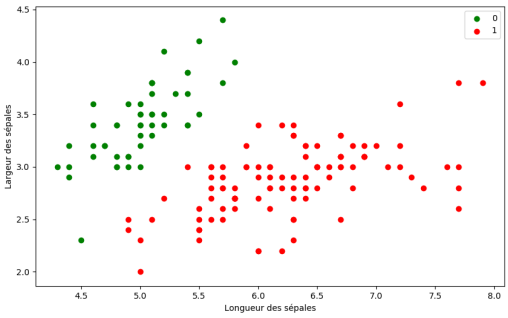

Figure 2 : visualisation du dataset Iris.

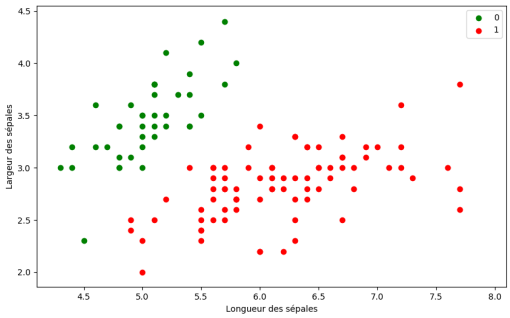

Figure 3 : échantillon d'apprentissage.

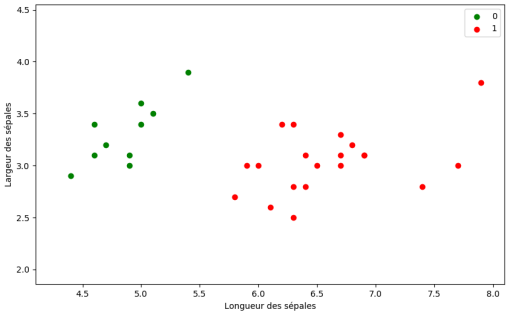

Figure 4 : échantillon test.

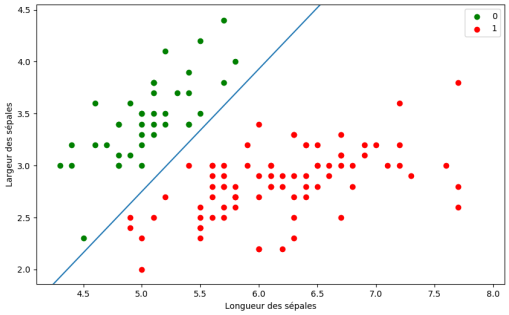

Figure 5 : l'hyperplan $\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$ sur l'échantillon d'apprentissage.

III. ANALYSE DISCRIMINANTE LINEAIRE

a) PRELIMINAIRE

On dispose d'un échantillon de n individus sur lequel on observe des variables explicatives $X_i = (X_i^1, \ldots, X_i^p)_{1 \le i \le n}$ et une variable qualitative $Y_i \in 1, \ldots, K$ qui correspond à chaque classe de l'individu i.

Notations :

- $C_1, \ldots, C_K$ sont les classes.
- $(\pi_1, \ldots, \pi_K)$ est la distribution de Y où π_K = P(Y=k) est la probabilité d'appartenir à la classe k et $\sum_{k=1}^{K} \pi_k = 1$.
- f_k est la densité conditionnelle du vecteur $X \epsilon R^n$ sachant que l'on est dans la classe k.
- $\mu_k \epsilon R^p$ est le vecteur des moyennes théoriques dans un cas gaussien.
- Σ_k est la matrice de variance-covariance théorique dans un cas gaussien.

b) PRINCIPE DE L'ANALYSE DISCRIMINANTE LINEAIRE

L'analyse discriminante linéaire a pour objectif d'expliquer et de prédire les valeurs d'une variable qualitative, notée Y, à partir de variables explicatives quantitatives et/ou qualitatives, notées $X = (X_1, \ldots . X_p)$.

Cette méthode s'appuie sur deux hypothèses :

- Les densités conditionnelles sont gaussiennes, c'est-à-dire que :

$$f_k(x) = \frac{1}{(\sqrt{2\pi})^p \sqrt{det\Sigma_k}} exp(-\frac{1}{2} * \quad {}^t(x-\mu_k)\Sigma_k^{-1}(x-\mu_k))$$

- L'homoscédasticité, c'est-à-dire qu'il y a égalité des matrices de variances-covariances conditionnelles :

$$\Sigma_1 = \ldots = \Sigma_K = \Sigma$$

L'ADL peut être présentée à travers l'approche bayesienne, qui consiste à modéliser la probabilité d'appartenance à une certaine classe et donc à lui affecter une nouvelle observation.

Formule de Bayes :

$$P(Y = k|X = x) = \frac{\pi_k \, f_k(x)}{\sum_{l=1}^{K} \pi_l \, f_l(x)}$$

Afin de trouver la probabilité conditionnelle $P(Y = k|X = x)$ la plus grande, une méthode consiste à chercher le maximum de vraisemblance :

$$\hat{Y} = \; argmax_{1 \le k \le K} \, P(Y = k|X = x) \iff \hat{Y} = argmax_{1 \le k \le K} \, \log(Y = k|X = x)$$

c) FONCTION LINEAIRE DISCRIMINANTE

Cette fonction est définie par :

$$\delta_k(x) = log(\pi_k) + \quad {}^t x.\Sigma^{-1}\mu_k - \frac{1}{2} * \quad {}^t\mu_k\Sigma^{-1}\mu_k$$

Elle est importante puisque :

$$\hat{Y} = \underset{1 \le k \le K}{\operatorname{argmax}} \, \delta_k(x)$$

Dans mon code, j'ai programmé cette fonction à l'intérieur de la méthode ***prediction*** qui se trouve dans la classe ***ADL***.

Preuve :

Comme $\sum_{l=1}^{K} \pi_l \, f_l(x)$ ne dépend pas de k, alors :

$$\hat{Y} = \underset{1\leq k\leq K}{\operatorname{argmax}}\ \pi_k f_k(x)$$

Ici :

$$\pi_k f_k(x) = \pi_k \frac{1}{(\sqrt{2\pi})^p \sqrt{det\Sigma}} exp(-\frac{1}{2} * \quad {}^t(x-\mu_k)\Sigma^{-1}(x-\mu_k))$$

En passant au logarithme, on obtient :

$$\log\pi_k f_k(x) = \log\pi_k - \frac{p}{2} log(2\pi) - \frac{1}{2}\log(det\Sigma) - \frac{1}{2} * \quad {}^t(x-\mu_k)\Sigma^{-1}(x-\mu_k)$$

Comme $\frac{p}{2} log(2\pi)$ et $\frac{1}{2} log(det\Sigma)$ sont constantes alors :

$$\hat{Y} = \underset{1\leq k\leq K}{\operatorname{argmax}}(\ log(\pi_k) - \frac{1}{2} * \quad {}^t(x-\mu_k)\Sigma^{-1}(x-\mu_k)\ )$$

En développant, on retrouve finalement :

$$\hat{Y} = \underset{1\leq k\leq K}{\operatorname{argmax}}(\ log(\pi_k) + \quad {}^t x.\Sigma^{-1}\mu_k - \frac{1}{2} * \quad {}^t\mu_k\Sigma^{-1}\mu_k\ )$$

d) ESTIMATION DES PARAMETRES

En pratique, on ne connaît ni les $(\pi_k)_{k=1}^K$, ni les $(\mu_k)_{i=1}^K$, ni Σ. Il faut donc les estimer par :

- $\hat{\pi}_k = \frac{n_k}{n}$ où $n_k = \sum_{i=1}^k \mathbb{I}_{Y_i=k}$
- $\hat{\mu}_k = \frac{1}{n_k}\sum_{i\in C_k} X_i$
- $\hat{\Sigma} = \frac{1}{n-K}\sum_{k=1}^K \sum_{i\in C_k}(X_i - \hat{\mu}_k)\ {}^t(X_i - \hat{\mu}_k)$ (Estimateur sans biais)

Dans mon code, ces expressions ont été codé en tant que méthodes dans la classe ***ADL*** sous les noms de ***pi***, ***mu*** et ***sigma*** respectivement.

e) APPLICATION

Dans cette section, j'analyse également la base de données historique —*Iris de Fisher*— dans le cas binaire. L'échantillon d'apprentissage et l'échantillon test sont les mêmes que ceux utilisés pour la régression logistique. Donc, les fleurs peuvent appartenir à un de ces deux groupes (0 ou 1).

Comme je l'ai dit précédemment, l'ADL repose sur deux hypothèses (les densités conditionnelles gaussiennes et l'homoscédasticité) qui devraient être respectées pour

appliquer cette méthode. Mais je les ai admis afin de travailler sur la même base de données et voir si mon code fonctionnait puisque le but principal est la programmation en python.

A l'aide de mon code, je trace la frontière de décision sur l'échantillon d'apprentissage (voir Figure 6) qui s'agit d'une droite. De plus, mon code m'a permis d'obtenir les résultats concernant les 30 fleurs dans une matrice de confusion. On peut constater que les étiquettes de ces fleurs ont été prédites correctement, ce qui prouve le bon fonctionnement de mon code.

	Valeurs réelles	
Valeurs prédites	1	0
1	10	0
0	0	20

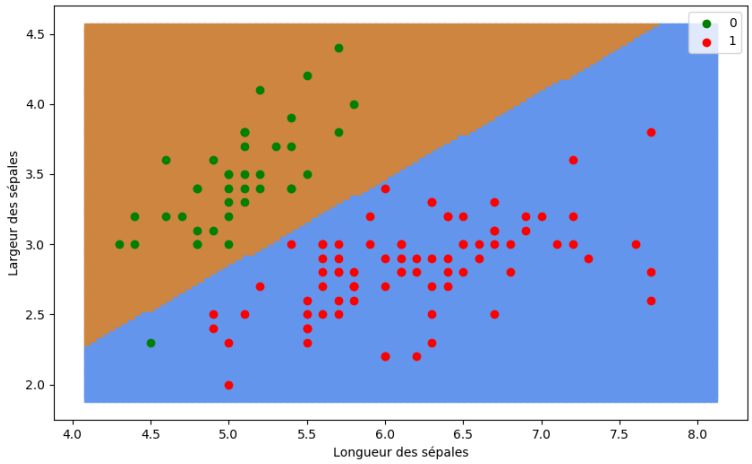

Figure 6 : Frontière de décision de l'ADL

IV. CONCLUSION

Tout au long de ce projet, j'ai fait plusieurs tâches qui m'ont permis de mieux comprendre la régression logistique et l'analyse discriminante linéaire, ainsi que les algorithmes utilisés pour chercher les estimateurs. Aussi, j'ai réussi à programmer ces deux techniques en python qui m'ont aidé à manipuler différents objets dans un langage que je connaissais depuis peu de temps. Il est vrai qu'il reste un vaste champ à approfondir telles que la significativité des modèles et l'étude de ces procédés en grande dimension que j'aurais éventuellement l'occasion de les coder dans un futur proche.