GIF 4101 Rendu 1 Rapport Arnaud DALIE

Arnaud Dalié

September 2018

1 Introduction

L'objectif de ce premier travail est de se familiariser avec les différentes méthodes paramétriques de base utilisées pour la classification. Ce compte rendu a pour but de compléter les fichier *d1q2.py* et *d1q2.py* en détaillant les choix effectués, les améliorations possibles ainsi que de répondre aux différentes questions de l'énoncé.

2 Travail à réaliser

2.1 Estimateurs statistiques

Cette partie a pour but d'étudier l'estimateur λ de la loi exponentielle grâce à la méthode du maximum de vraissemblace

a) On a :

$$p(x) = \begin{cases} \lambda exp(-\lambda x)\ x \in [0; \infty[\\ 0\ x \in]\infty; 0] \end{cases}$$

On calcule la fonction de vraissemblace :

$$l(\lambda|X) = \prod_{i=1}^{n} p(x_i|\lambda)$$

On remarque que si ne serait-ce qu'une seule valeur de X est négative, alors la vraissemblance tombe à 0. C'est pourquoi on admettra que les valeurs de X sont toutes positives.

$$l(\lambda|X) = \prod_{i=1}^{n} \lambda e^{\lambda x_i}$$

Ensuite, on calcule la log-vraissemblance et on obtient :

$$L(\lambda|X) = \sum_{i=1}^{n} ln(\lambda) - \lambda x_i$$

$$L(\lambda|X) = nln(\lambda) - \lambda \sum_{i=1}^{n} x_i$$

On va chercher la valeure de λ pourl aquelle cette valeure est maximale simplement en cherchant le point où la dérivée s'annule :

$$\frac{\partial L(\lambda|X)}{\partial \lambda} = \frac{n}{\lambda} - \sum_{i=1}^{n} x_i$$

$$\frac{\partial L(\lambda|X)}{\partial \lambda} = 0 \Leftrightarrow \frac{n}{\lambda} = \sum_{i=1}^{n} x_i$$

$$\Leftrightarrow \frac{1}{\lambda} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

$$\Leftrightarrow \lambda = \frac{1}{\bar{x}}$$

On vérifie aisément que la dérivée seconde est négative pour tous $\lambda > 0$ et donc qu'on a bien à faire a un extremum qui est un maximum.

b) On va maintenant chercher à savoir si cet estiamteur est sans biais. On cherche donc à vérifier :

$$b_\lambda(\hat{\lambda}) = 0$$

$$\Leftrightarrow E_X\left[\hat{\lambda}(X)\right] - \lambda \Leftrightarrow E\left[\frac{1}{\bar{x}}\right] - \frac{1}{E\left[X\right]}$$

$$\Leftrightarrow E\left[\frac{1}{\bar{x}}\right] = \frac{1}{E\left[E\left[X\right]\right]}$$

$$\Leftrightarrow E\left[\frac{1}{\bar{x}}\right] = \frac{1}{E\left[\bar{x}\right]}$$

Or, on sait que :

$$E\left[\frac{1}{x}\right] \neq \frac{1}{E\left[x\right]}$$

On peut donc conclure que l'estiamteur a bien un biais.

2.2 Expérimentations avec scikit-learn

Dans cette partie, on va tester les performances de dofférents classificateurs sur le jeu de données des Iris de Fisher.

a) Voici les différentes variables mesurées par Fisher comparée deux à deux avec les classes représentées dans différentes couleurs. On remarque que les différentes sont facilement distingables pour les différents couples de variables étudiées, à l'exception de la classe verte et rouge dans le cas de la comparaison entre sepal width et sepal length (figure en haut à gauche). On constate également la nette séparation de la classe bleu avec les deux autres.

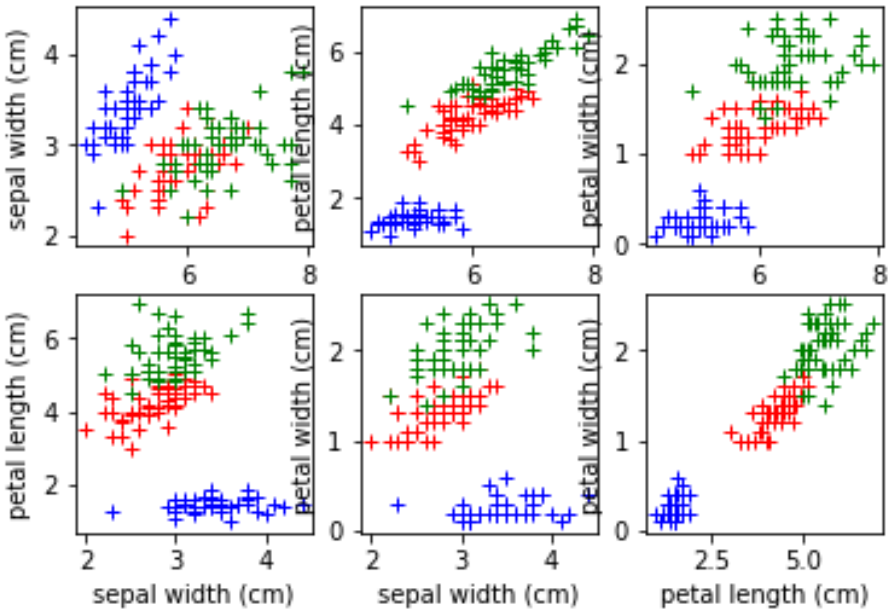

Figure 1: Comparison of classes ratios between different variables

b) On teste les différents classifieurs reposant sur l'Annalyse Discriminante Quadratique, l'Annalyse Discriminante Linéaire, la classification Bayesienne naïve et enfin la distance aux centroïdes. Voici le tableau résumant les performances de chacuns pour les différents couples de variables.

	Quadratic	Linear	Naïve Bayesian	Centroïd	Mean
SW / SL	0.200	0.200	0.220	0.185	0.201
PL / SL	0.040	0.033	0.087	0.113	0.188
PW / SL	0.033	0.040	0.040	0.060	0.043
PL / SW	0.047	0.047	0.080	0.060	0.059
PW / SW	0.047	0.033	0.053	0.060	0.048
PW / PL	0.020	0.040	0.040	0.040	0.035

On remarque que les performances sont croissantes en fonction de la complexité du modèle. Avec ce jeu de données, on peut considérer le modèle des centroïdes comme un bon compromis entre performance et complexité ($\approx 90\%$). Il a, de manière surprenante, la meilleure performance pour le couple de variable qu'on avait considéré comme difficile à distinguer.

c) Pour cette question, on ne sépare plus les variables en couples. On fait directement l'entraînement et les tests sur les 4 variables à la fois. Cela rend la tâche plus facile pour le classifieur et on obtient pour les trois méthodes

de test les resultats suivants : $err = 0.020$ quand on utilise le même ensemble d'apprentissage et de test, $err = 0.033$ pour l'utilisation de la moitié de l'ensemble pour l'apprentissage et enfin $err = 0.024$ pour la méthode à $k = 3$ plis. On pouvait s'attendre à ces résultats. Le premier semble avoir la meilleur performance, mais on a surtout à faire à un $overfitting$ puisque les données utilisées pour apprendre le modèle sont les mêmes que pour le tester. Ensuite, on remarque que la méthode à $k = 3$ plis est plus performante que la seconde méthode. Pour une séparation en deux sous-ensembles égales, le modèle dispose de moins de données d'apprentissage que pour $k = 3$ dans la troisième méthode. On peut donc supposer qu'il y à un k optimal pour avoir un maximum de données d'entraînement sans tomber dans l'overfitting.

d) On reteste les quatre classifieurs avec un jeu de données formant des cercles concentriques bien distinct. Comme on pouvait s'y attendre, les classifieurs linéaires comme LDA et NearestCentroids n'ont pas pu décider linéairement des classes et affichent donc des taux d'erreurs en moyenne de 50% mais cependant très variables elon les conditions initiales. (Par exemple, vu que les deux centres sont quasiment identiques, python a du mal à calculer la pente de la frontière de décision et elle change donc à chaque execution. Ce calcul obéit sans doute à des lois d'arrondis et des approximations très bas niveau et hors du programme de ce cours)

2.3 Classement avec option de rejet

Le but de cette dernière partie est l'implémentation d'un classificateur avec option de rejet. Avant de se lancer dans l'implémentation, il était demandé un petit exercice théorique consistant à calculer l'estimateur de σ puis à expliciter la règle de décision avec rejet.

a) Les calculs suivent la même méthode que dans la partie 1.

$$l(\sigma_i|X_i) = \prod_{t=1}^{p} p(x_i^t|\sigma_i)$$

$$= \prod_{t=1}^{p} \frac{1}{\sqrt{2}\pi^D} \frac{1}{\sqrt{d}et(\Sigma_i)} e^{-\frac{1}{2}(x-\mu)^T \Sigma_i^{-1}(x-\mu)}$$

$$= \prod_{t=1}^{p} (2\pi)^{\frac{-N}{2}} det(\Sigma_i)^{-\frac{1}{2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma_i^{-1}(x-\mu)}$$

La matrice Σ étant de la forme $\sigma^2 I$, on a :

$$\Sigma_i^{-1} = \frac{1}{\sigma_i^2} I$$

et :

$$det(\Sigma_i) = \sigma i^{2D}$$

On obtient donc la log-vraissemblance suivante :

$$L(X_i|\sigma_i) = \sum_{t=1}^{p} -\frac{D}{2} ln(2\pi) - D ln(\sigma_i) - \frac{1}{2\sigma_i^2} \|x_t - \mu\|^2$$

$$= -\frac{Dp}{2} ln(2\pi) - Dp ln(\sigma_i) - \frac{1}{2\sigma_i^2} \sum_{t=1}^{p} \|x_t - \mu\|^2$$

En dérivant :

$$\frac{\partial^2 L(X_i|\sigma_i)}{\partial \sigma_i^2} = -\frac{Dp}{\sigma_i} + \frac{1}{\sigma_i^3} \sum_{t=1}^{p} \|x_t - \mu\|^2$$

On cherche alors pour quelle valeur de σ la dérivé s'annule :

$$\frac{\partial^2 L(X_i|\sigma_i)}{\partial \sigma_i^2} = 0$$

$$\Leftrightarrow \frac{Dp}{\sigma_i} = \frac{1}{\sigma_i^3} \sum_{t=1}^{p} \|x_t - \mu\|^2$$

$$\Leftrightarrow \sigma_i^2 = \frac{1}{Dp} \sum_{t=1}^{p} \|x_t - \mu\|^2$$

$$\Leftrightarrow \sigma_i = \sqrt{\frac{1}{Dp}\sum_{t=1}^{p} \|x_t - \mu\|^2}$$

b) La probabilité à posteriori $P(C_i|X)$ peut donc être calculée. La formule développée est la suivante :

$$P(C_i|x) = \frac{P(C_i) * P(x|C_i)}{P(x)} = \frac{P(C_i) * P(x|C_i)}{\sum_{i=1}^{N} P(C_i) * P(x|C_i)}$$
$$= \frac{\frac{P_i}{P} * \frac{1}{\sqrt{2\pi\sigma_i^{2D}}} e^{\frac{1}{2\sigma_i^2}\|x-\mu_i\|^2}}{\sum_{j=1}^{N} \frac{P_j}{P} * \frac{1}{\sqrt{2\pi\sigma_j^{2D}}} e^{\frac{1}{2\sigma_j^2}\|x-\mu_j\|^2}}$$

La fonction de perte s'écrit :

$$L(c_i, C_j) = \begin{cases} si\ i = j \\ 1\ si\ i \neq j \\ \lambda\ si\ i = K+1 \end{cases}$$

Le risque s'écrit alors comme suit pour les classes:

$$R(c_i|x) = \sum_{k=1}^{K} L(c_i, C_j) P(C_k|x) = 1 - P(C_i|x)$$

pour l'une des classes. Et le risue de l'option de rejet s'écrit alors :

$$R(c_{K+1}|x) = \sum_{k=1}^{K} \lambda P(C_k|x) = \lambda$$

En résolvant l'inéquation du risque, on obtient la fonction de prise décision minimisant le risque en fonction de λ :

$$c_i = \begin{cases} argmax_{C_j=C_1}^{C_N}(P(C_j|x))\ si\ max(P(C_j|x)) > 1 - \lambda \\ C_{N+1}\ sinon \end{cases}$$

c) L'implémentation du classifieur ave coption de rejet se fait en quatres fonctions :
- fit : qui estime les différents paramètres du modèle pour chaque classe en fonction de l'ensemble donné.
- predict : qui calcule la probabilité de chaque classe en fonction des paramètres préalablement calculés.
- predict_proba : qui applique la fonction de décision afin de choisir la classe prédite.
- score : qui appelle predic_proba et retourne le taux de décision correctes.

d) Les performances pour les différents λ sont les suivantes :

λ	0.1	0.3	0.5	1
SW / SL	0.607	0.387	0.200	0.193
PL / SL	0.233	0.147	0.113	0.113
PW / SL	0.36	0.22	0.133	0.133
PL / SW	0.200	0.107	0.053	0.053
PW / SW	0.260	0.80	0.600	0.600
PW / PL	0.107	0.073	0.033	0.033

A chaque fois, c'est le classifieur acceptant ne rejetant rien qui obtient le meilleur score. L'erreur est même décroissante avec l'augmentation de λ. On peut expliquer cela par le bon groupement et la bonne séparation des classes. En effet, les données sont bien regroupées et nes chevauchent quasiment pas (sauf pour une paire de variables). De fait, très peu de données ont des probabilités inférieures au seuil fixé par *lambda*.

On peut conclure que dans les λ choisis, aucun ne permet d'obtenir de resultats meilleurs sur notre dataset. Attention cependant, il se peut que sur de nouvelles données aberrantes ou plus bruitées, les régions de rejet soit plus déterminantes.