Université de Caen **Analyse de données avancée**
UFR des Sciences Examen TP du 14 décembre 2017 (rattrapage). Durée 2h par F. Chamroukhi
M2 Statistique/Informatique Seuls le cours et les travaux personnels sont autorisés 2017/2018

Consignes : Vous devez séparer pour chacun des exercices votre programme principal (l'équivalent du main) du reste du code (fonctions etc). Le dépôt de votre travail se fera sous la forme d'une archive (qui portera votre nom et qui contiendra également les données) à déposer sur ecampus (exclusivement) le 14/12/2017 avant 16h00, heure stricte.

Barème : Le barème est donné à titre indicatif. Les exercices pour lesquels les codes ne s'exécutent pas, seront notés sur la moitié des points du barème.

Ex1 : Analyse supervisée (10 pts) :

On considère un échantillon indépendant $((\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n))$ d'individus $\boldsymbol{X}_i$ décrits par d variables réelles ($\boldsymbol{X}_i \in \mathbb{R}^d$) d'une population de K classes hétérogènes telle que $Y_i \in [\![1, K]\!]$ est la classe de l'individu $\boldsymbol{X}_i$. On dispose d'un échantillon observé datatrain-bis $((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n))$. L'objectif est de prédire les classes pour un échantillon de test issu de cette même population sur la base d'un modèle probabiliste paramétrique de paramètres $\boldsymbol{\theta}$. La prédiction peut ainsi s'effectuer par la règle du maximum a posteriori (MAP) qui consiste à maximiser la probabilité a posteriori pour avoir une prédiction $\widehat{y}_i$ de la classe de l'individu $\boldsymbol{x}_i$, étant donné un modèle de paramètres $\hat{\boldsymbol{\theta}}$ appris à partir de l'échantillon observé :

$$\widehat{y}_i = \arg \max_{y_i \in [\![1,K]\!]} \mathbb{P}(Y_i = y_i | \boldsymbol{X} = \boldsymbol{x}_i; \boldsymbol{\theta}). \tag{1}$$

Modélisation discriminative : Régression logistique : Une méthode probabiliste discriminative recherche directement un modèle de la loi conditionnelle $\mathbb{P}(Y|\boldsymbol{X})$ et s'intéresse donc prioritairement à séparer les classes. C'est le cas de la régression logistique qui s'appuie sur un modèle linéaire généralisé de la forme suivante pour les loi conditionnelles

$$\mathbb{P}(Y = k | \boldsymbol{X} = \boldsymbol{x}; \boldsymbol{\theta}) = \frac{\exp(\beta_{k0} + \boldsymbol{\beta}_k^T \boldsymbol{x})}{1 + \sum_{\ell=1}^{K-1} \exp(\beta \ell 0 + \boldsymbol{\beta}_\ell^T \boldsymbol{x})} \tag{2}$$

pour $k \in [\![1, K-1]\!]$ et $\mathbb{P}(Y = K | \boldsymbol{X} = \boldsymbol{x}; \boldsymbol{\theta}) = \frac{1}{1+\sum_{\ell=1}^{K-1} \exp(\beta \ell 0 + \boldsymbol{\beta}_\ell^T \boldsymbol{x})}$.
Cela correspond donc à une transformation par une fonction exponentielle normalisée (appelée aussi softmax) d'une fonction linéaire en $\boldsymbol{x}$. Cela explique en effet le fait que en régression logisitique on cherche à modéliser les probabilités a posteriori des classes à travers des fonctions linéaires en $\boldsymbol{x}$ puisque que l'on a $\log\left(\frac{\mathbb{P}(Y=k|\boldsymbol{X}=\boldsymbol{x})}{\mathbb{P}(Y=K|\boldsymbol{X}=\boldsymbol{x})}\right) = \beta_{k0} + \boldsymbol{\beta}_k^T \boldsymbol{x}$ pour tout $k \in [\![1, K-1]\!]$.
Le vecteur paramètre du modèle défini par $\boldsymbol{\theta} = (\boldsymbol{\beta}_1^T, \ldots, \boldsymbol{\beta}_{K-1}^T)^T \in \mathbb{R}^{(K-1)\times(d+1)}$ avec $\boldsymbol{\beta}_k = (\beta_{k0}, \boldsymbol{\beta}_k^T)^T$ est estimé en maximisant la log-vraisemblance conditionnelle par rapport à $\boldsymbol{\theta}$:

$$\begin{aligned}
\log L(\boldsymbol{\theta}; \mathbf{X}, \boldsymbol{y}) &= \log \prod_{i=1}^{n} \mathbb{P}(Y_i | \boldsymbol{X}_i = \mathbf{x}_i; \boldsymbol{\theta}) \\
&= \log \prod_{i=1}^{n} \prod_{k=1}^{K} \mathbb{P}(Y_i = k | \boldsymbol{X}_i = \boldsymbol{x}_i; \boldsymbol{\theta})^{y_{ik}} \\
&= \sum_{i=1}^{n} \sum_{k=1}^{K} y_{ik} \log \mathbb{P}(Y_i = k | \boldsymbol{X}_i = \boldsymbol{x}_i; \boldsymbol{\theta})
\end{aligned} \tag{3}$$

où y_{ik} est une variable binaire telle que $y_{ik} = 1$ si et seulement si $y_i = k$ (i.e, $\boldsymbol{x}_i$ appartient à la classe k). Cette log-vraisemblance ne peut pas être maximisée analytiquement et pour ce faire on utilise l'algorithme Newton-Raphoson.

On considère un problème à deux classes ($Y_i \in [\![0, 1]\!]$, i.e., régression logistique binaire). Dans ce cas le modèle est défini par les probabilités

$$\pi(\boldsymbol{x}; \boldsymbol{\theta}) = \mathbb{P}(Y = 1 | \boldsymbol{X} = \boldsymbol{x}; \boldsymbol{\theta}) = \frac{\exp(\beta_{10} + \boldsymbol{\beta}_1^T \boldsymbol{x})}{1 + \exp(\beta_{10} + \boldsymbol{\beta}_1^T \boldsymbol{x})} \tag{4}$$

et $\mathbb{P}(Y = 0 | \boldsymbol{X} = \boldsymbol{x}; \boldsymbol{\theta}) = 1 - \pi(\boldsymbol{x}; \boldsymbol{\theta})$.
Le vecteur paramètre du modèle défini par $\boldsymbol{\theta} = (\beta_0, \boldsymbol{\beta}_1^T)^T \in \mathbb{R}^{d+1}$ est estimé en maximisant par rapport à $\boldsymbol{\theta}$ la log-vraisemblance conditionnelle qui dans ce cas de classification binaire prend la forme :

$$\begin{aligned}
\log L(\boldsymbol{\theta}; \mathbf{X}, \boldsymbol{y}) &= \log \prod_{i=1}^{n} \mathbb{P}(Y_i = 1 | \boldsymbol{X}_i = \boldsymbol{x}_i; \boldsymbol{\theta})^{y_i} \mathbb{P}(Y_i = 0 | \boldsymbol{X}_i = \boldsymbol{x}_i; \boldsymbol{\theta})^{1-y_i} \\
&= \sum_{i=1}^{n} y_i \log \mathbb{P}(Y_i = 1 | \boldsymbol{X}_i = \boldsymbol{x}_i; \boldsymbol{\theta}) + (1 - y_i) \log \mathbb{P}(Y_i = 0 | \boldsymbol{X}_i = \boldsymbol{x}_i; \boldsymbol{\theta}) \\
&= \sum_{i=1}^{n} y_i \log \pi(\boldsymbol{x}_i; \boldsymbol{\theta}) + (1 - y_i) \log (1 - \pi(\boldsymbol{x}_i; \boldsymbol{\theta})) \\
&= \sum_{i=1}^{n} y_i (\beta_{10} + \boldsymbol{\beta}_1^T \boldsymbol{x}_i) - \log(1 + \exp(\beta_{10} + \boldsymbol{\beta}_1^T \boldsymbol{x}_i)).
\end{aligned} \tag{5}$$

On montre que dans le cas de ce modèle, l'algorithme de Nerwton-Raphson consiste à partir d'un modèle initial de paramètres $\boldsymbol{\theta}^{(0)}$ et à mettre à jour, à chaque itération t, les paramètres selon l'équation de mise à jour suivante, jusqu'à ce qu'il n'y ait plus d'augmentation significative au sens d'un seuil préfixé de la log-vraisemblansce conditionnelle (3) :

$$\begin{aligned}
\boldsymbol{\theta}^{(t+1)} &= \boldsymbol{\theta}^{(t)} + (\mathbf{X}^T \mathbf{W}^{(t)} \mathbf{X})^{-1} \mathbf{X}^T (\boldsymbol{y} - \boldsymbol{p}^{(t)}) \\
&= (\mathbf{X}^T \mathbf{W}^{(t)} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W}^{(t)} \widetilde{\boldsymbol{y}}
\end{aligned} \tag{6}$$

où :

- $\mathbf{X}$ est la matrice de dimensions $n \times (d + 1)$ de lignes $(1, \boldsymbol{x}_i^T)$ pour $i = 1, \ldots, n$;
- $\boldsymbol{y}$ est le vecteur colonne de dimension $n \times 1$ des labels y_i : $\boldsymbol{y} = (y_1, \ldots, y_n)^T$;
- $\boldsymbol{p}$ est le vecteur colonne de dimension $n \times 1$ des probabilités : $\boldsymbol{p} = (\pi(\boldsymbol{x}_1; \boldsymbol{\theta}), \ldots, \pi(\boldsymbol{x}_n; \boldsymbol{\theta}))^T$ avec $\pi(\boldsymbol{x}_i; \boldsymbol{\theta}) = \mathbb{P}(Y_i = 1 | \boldsymbol{x}_i; \boldsymbol{\theta}) = \frac{\exp(\boldsymbol{\beta}_k^T \boldsymbol{x}_i)}{1 + \exp(\boldsymbol{\beta}_k^T \boldsymbol{x}_i)}$;
- $\mathbf{W}$ est la matrice diagonale de dimension $n \times n$ d'élements $\pi(\boldsymbol{x}_1; \boldsymbol{\theta}) (1 - \pi(\boldsymbol{x}_n; \boldsymbol{\theta}))$: $\mathbf{W} = \mathrm{diag}(\boldsymbol{p})$.
- $\widetilde{\boldsymbol{y}} = \mathbf{X} \boldsymbol{\theta}^{(t)} + (\mathbf{W}^{(t)})^{-1} (\boldsymbol{y} - \boldsymbol{p}^{(t)})$

De part la forme de (6) qui correspond à celle de l'estimateur des moindres carrés pondérés (avec une matrice de poids $\mathbf{W}$) on appelle l'algorithme Netwton-Raphson dans ce cas l'algorithme IRLS (Iterative Reweighted Least Squares pour moindres carrés pondérés itératifs).

Travail demandé :

1. Implémenter le et tester le sur l'échantillon de datatest-bis issus d'une même population de K classes hétérogènes de données de dimension d. Pour cela, créer une fonction `train_LogReg.m` pour l'apprentissage et une fonction `predict_LogReg.m` pour la prédiction.
2. Calculer le taux d'erreur de prédiction pour l'échantillon de test étiqueté `datatest`.
3. Maintenant considérer l'échantillon AllXtest et prédire ses classes
4. Afficher, sur le même graphique, les données d'apprentissage et les données de test classées
5. Comparer vos résultats à ceux obtenus en se basant sur les fonctions `mnrfit` et `mnrval` de Matlab.
6. Comparer vos résultats à ceux obtenus en se basant sur les fonctions `glmfit` et `glmval` de Matlab.

Ex2 : Analyse non-supervisée (10 pts) :

On dispose d'un n-échantillon $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ issu d'une population hétérogène de K classes d'individus binaires multivariés décrits par d variables indépendantes : $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{id})^T \in \{0,1\}^d$ et distribués selon une loi mélange de lois de Bernoulli multivariées définie par :

$$\mathbb{P}(\boldsymbol{X}_i = \boldsymbol{x}_i; \boldsymbol{\theta}) \quad = \quad \sum_{k=1}^{K} \pi_k \, \mathcal{B}(\boldsymbol{x}_i; \boldsymbol{p}_k) \tag{7}$$

où $\mathcal{B}(\boldsymbol{x}_i; \boldsymbol{p}_k) = \prod_{j=1}^{d} \mathcal{B}(x_{ij}; p_{kj})$ est la loi de Bernoulli multivariée de la classe k avec $\mathcal{B}(x_{ij}; p_{kj})$ la loi de Bernoulli univariée de paramètre $p_{kj} \in]0,1[$ associée à la variable x_{ij} et définie par $\mathcal{B}(x; p) = p^x(1-p)^{1-x}$.

Afin d'apprendre ce modèle en estimant les paramètres $\boldsymbol{\theta} = (\pi_1, \ldots, \pi_K, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_K)$ à partir des données et faire la classification non-supervsiée des données, on utilise l'algorithme Espérance-Maximisation (EM). On montre que pour ce modèle l'algorithme EM consiste à partir d'un modèle initial de paramètres $\boldsymbol{\theta}^{(0)}$ et alterner à chaque itération t entre les deux étapes E- et M- suivantes jusqu'à ce qu'il n'y ait plus d'augmentation significative au sens d'un seuil préfixé de la log-vraisemblance du modèle :

1. Étape E : Calculer les probabilités a posteriori : $\tau_{ik}^{(t)} = \mathbb{P}(Y_i = k | \boldsymbol{x}_i, \boldsymbol{\theta}^{(t)}) = \frac{\pi_k^{(t)} \, \mathcal{B}(\boldsymbol{x}_i; \boldsymbol{p}_k^{(t)})}{\sum_{\ell=1}^{K} \pi_\ell^{(t)} \, \mathcal{B}(\boldsymbol{x}_i; \boldsymbol{p}_\ell^{(t)})}$

2. Étape M : Mettre à jour les paramètres du modèle $\boldsymbol{\theta}^{(t+1)}$: $\pi_k^{(t+1)} = \frac{\sum_{i=1}^{n} \tau_{ik}^{(t)}}{n}$ et $\boldsymbol{p}_k^{(t+1)} = \frac{\sum_{i=1}^{n} \tau_{ik}^{(t)} \boldsymbol{x}_i}{\sum_{i=1}^{n} \tau_{ik}^{(t)}}$.

Travail demandé :

1. Implémenter l'algorithme EM pour estimer les paramètres $\boldsymbol{\theta}$ du modèle (7) et les classes $\boldsymbol{y} = (y_1, \ldots, y_n)$ des données Xbin sur la base de ce modèle avec la règle du MAP (1). On pourra fixer la valeur de K à 3.
2. Affiche les valeurs de la log-vraisemblance enregistrée au cours de l'algorithme et les données classées. Pour ce dernier graphique, on utilisera la commande `imagesc(X(sort(y),:))` où `X` sont les données et `y` les classes obtenues.
3. Utiliser le critère BIC pour sélectionner le nombre de classes K sachant que $K \in [\![1, 10]\!]$