A2DI - TP n°5

Dans ce cinquième TP, nous allons nous attaquer à notre 1er problème supervisé multi-classes sur un jeu de données réelles. Ce jeu de donnée s'appelle `20newsgroup`. Il se compose d'exemples $\mathbf{x}^{(i)}$ qui indique la présence de certains mots dans des articles publiés sur des sites d'actualité.

Les classes à reconnaître sont les sujets généraux des documents :

1. *computer science*,
2. *recap*, (compte-rendu sportif)
3. *science*,
4. *talk* (forum de discussions politiques).

Voici, pour exemple, un extrait d'un article de sport publié dans le New York Times :

> ***Andy Murray Beats Error-Prone Novak Djokovic to Retain No. 1 Ranking***
> *"... In the first half of the season, Djokovic had looked nearly invincible, squelching Murray and everyone else on the sport's biggest stages.*
> *He had won three of the two players' four meetings during that time, in the finals of the Australian Open, the French Open and the Madrid Open. Murray notched a straight-set win in the title match in Rome..."*

On observe que certains mots sont caractéristiques de la classe 2 (*win, match, player ..*). Si ces mots sont sélectionnés pour former un attribut $x_j^{(i)}$ des vecteurs $\mathbf{x}^{(i)}$, on s'attend donc à avoir $x_j^{(i)} = 1$ si $\mathbf{x}^{(i)}$ appartient à la classe 2, ($c^{(i)} = 2$).

Supposons que $\dim(\mathbb{X}) = 4$ et que les mots que nous recherchons sont $\{win, match, player, computer\}$. Cet ensemble est appelé **dictionnaire**. Nous allons pouvoir "vectoriser" le document textuel en le représentant par le vecteur de bits suivant :

$$\mathbf{x} = \begin{pmatrix} win & match & player & computer \\ 1 & 1 & 1 & 0 \end{pmatrix}. \tag{1}$$

Le choix des mots qui constituent le dictionnaire est très important mais ne sera pas abordé aujourd'hui. Nous partons donc d'un ensemble de mots pré-sélectionnés. Nous utiliserons un classifieur naïf bayésien pour résoudre ce problème à 4 classes. La "vectorisation" de documents textuels selon le principe décrit ci-dessus est appelée "**sac de mots**" (*bag of words*).

Exercice n°1 : Classifieur naïf bayésien et sac de mots

Dans cet exercice, nous allons utiliser le dataset `20newsgroup` fourni sur la page personnelle de Sam Roweis (NYU). Il fournit sous-ensemble des données complètes.

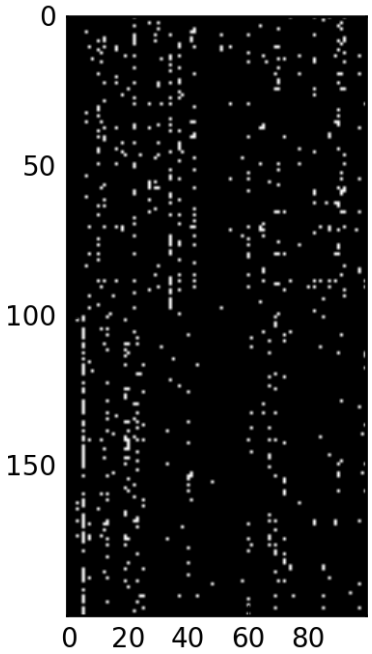
Questions :

1. Téléchargez le fichier `20news_w100.mat` qui contient le dataset. Ensuite chargez ces données et obtenez des variables informatiques correspondant comme d'habitude aux exemples d'apprentissage et aux classes :

```
data = scipy.io.loadmat('20news_w100.mat')
X=data["documents"].toarray()
c=data["newsgroups"][0]-1
```

On prend la convention que la numérotation des classes commence à 0.

2. Insérez dans une variable `n` le nombre d'exemples, dans `d` la dimension des exemples et dans `n_class` le nombre de classe.
3. Créez une fonction `kfold_data` qui découpe un dataset selon le principe de la validation croisée à k plis. Cette fonction prend en entrée :
   — une variable `X` contenant des exemples d'apprentissage rangés selon le mode usuel (1 colonne par exemple, 1 ligne par dimension de $\mathbb{X}$)
   — une variable `c` contenant les classes des exemples contenus dans `X`,
   — l'entier k,
   — une variable `n_class` représentant le nombre de classes.
   Elle retourne 4 `tuples` de taille k. Le $i^{\text{ème}}$ élément de chacun de ces `tuples` contient respectivement, les exemples pour le train, les classes des exemples pour le train, les exemples pour le test et les classes des exemples pour le test correspondant au $i^{\text{ème}}$ pli de la validation croisée.
   Remarque : Vous pouvez bien sûr reprendre le code de la fonction `kfold_data` que nous avions écrite au TP précédent. Néanmoins, notre tâche de classification concerne aujourd'hui plus de 2 classes, il faudra donc calculer la proportion de chaque classe à copier dans les plis afin d'obtenir un panachage équilibré des classes. Vous appliquerez la fonction `kfold_data` à nos données avec $k = 5$.
4. Avant de commencer l'apprentissage, on souhaite visualiser une partie des données. Pour le 1^{er} pli, prélever les 100 premiers exemples de la classe 0 et les 100 premiers de la classe 1. Concaténez verticalement ces deux matrices en une seule (fonction `numpy.vstack`) et affichez cette matrice à l'aide de `matplotlib.pyplot.imshow`.
   Voici le résultat souhaité :



   Observez-vous un *pattern* ?
5. Nous allons utiliser un classifier naïf Bayésien sur ces données. Ce modèle repose sur la factorisation suivante de la distribution jointe :

$$p_{\mathbf{X},Y}(\mathbf{x}, c) = p_Y(c) \prod_{j=1}^{d} p_{X_j|Y=c}(x_j). \tag{2}$$

   Dans cette question, on souhaite estimer les probabilités des classes $p_Y(c)$. Nous utliserons la méthode statistique du *maximum likelihood estimate* (MLE). La distribution des classes suit une loi catégorique $\text{Cat}(\boldsymbol{\pi})$. Chaque entrée du vecteur $\boldsymbol{\pi}$ corresopond à une de ces probabilités : $p_Y(c) = \pi_c$. On a vu en

cours que le MLE pour ces paramètres s'obtient par simple dénombrement :

$$\hat{\pi}_{c,MLE} = \frac{\text{nombre de membres de la classe } c \text{ dans l'ensemble d'apprentissage}}{\text{taille de l'ensemble d'apprentissage}} \tag{3}$$

Calculez et enregistrez ces probabilités dans un `numpy array` (pour chaque pli).

6. Pour completer le modèle Bayésien naïf, il nous manque des estimés MLE des probabilités conditionnelles $p_{X_j|Y=c}(x_j)$. Chaque variable aléatoire X_j est binaire. Le modèle paramétrique pour de telles variables est la loi de Bernouilli : $X_j \sim \text{Ber}(\theta_{jc})$.
   Le MLE pour une loi de Bernouilli s'exprime également sous forme de dénombrement :

$$\hat{\theta}_{jc,MLE} = \frac{\text{nombre d'exemples de la classe } c \text{ dont le texte contient le mot N°} j}{\text{nombre d'exemples de la classe } c} \tag{4}$$

   Calculez et enregistrez ces probabilités dans un `2D numpy array` (pour chaque pli). La taille de ces tableaux est $d \times n\_\text{class}$.

7. Une fois l'estimation des paramètres achevée, l'apprentissage est terminé. On va maintenant passer à la phase de test. Calculez le taux d'erreur moyen sur les 5 plis.

8. L'estimation des paramètres du NBC par MLE est purement fréquentiste. Si nous avons des connaissance *a priori* sur les valeurs des paramètres θ_{jc} et π_c, on peut utiliser une approche Bayésienne et obtenir des estimés MAP (maximum *a posteriori*). Cela s'avère surtout intéressant que le nombre d'exemples d'apprentissage n n'est pas très grand devant le nombre de paramètres à apprendre.
   Dans ce TP, on se propose de calculer des estimés MAP pour les paramètres θ_{jc} seuleument. Si on suppose que nos connaissances *a priori* sont très limitées, on peut se contenter de supposer que toute valeur des paramètres est équi-probable, d'où :

$$\theta_{jc} \sim \mathcal{U}([0;1]), \tag{5}$$

   où $\mathcal{U}([0;1])$ représente la loi uniforme sur l'intervalle $[0;1]$. On montre alors que

$$\hat{\theta}_{jc,MAP} = \frac{\text{nombre d'exemples de la classe } c \text{ dont le texte contient le mot N°} j + 1}{\text{nombre d'exemples de la classe } c + 2}. \tag{6}$$

   Cette méthode d'estimation est aussi utilisée par les fréquentistes mais au simple motif (purement pragmatique) qu'elle évite d'avoir des probabilités nulles dans la factorisation du NBC. On parle de lissage de Laplace ou de *add-one-smoothing*.

   Implémentez cette estimation. Vous devez constater un écart insignifiant avec ceux obtenus par MLE.

9. Divisez la taille de l'ensemble d'apprentissage par 10 tout en conservant intacte celle de l'ensemble de test. Relancez les deux méthodes. Vous devez constater cette fois un écart de performance significatif en faveur de la méthode Bayésienne.