Les données à utiliser pour ce travail sont disponibles sur Moodle du cours. Ces données sont issues de différents domaines applicatifs (images, médical, astronomie, ...). Certaines ont été transformées ou peuvent être transformées en des problèmes de classification binaire.

Certaines données sont fournies sous forme de fichiers au format Matlab. Chaque fichier contient une matrice des données d'entrées (souvent notée $X \in \mathbb{R}^{N \times d}$) où N représente le nombre de points et d la dimension de chaque point. Chaque fichier contient également un vecteur souvent $Y \in \mathbb{R}^N$ ou `labels` correspondant aux labels des points. Dans certains fichiers, il y a la variable `datainfo` qui est un lien vers un site web décrivant en détails les données. Pour information, l'ensemble des données est issu de `http://archive.ics.uci.edu/ml/datasets.html`.

D'autres données sont fournies dans d'autres formats : par exemple les données `bank.csv` sont dans un fichier .csv à dont il faut extraire les matrices X et Y. Les données FMA [1] [2], analyse de morceaux de musique, nécessitent Python.

1. Contacter l'enseignant référent pour choisir un jeu de données. Pour certains problèmes (classification d'images, données astronomiques), des détails supplémentaires sont disponibles sur les pages 2 et 3 de cet énoncé.
2. Utiliser la technique de votre choix pour résoudre le problème.
3. Dans un rapport faisant au maximum 5 pages, décrire et justifier la méthodologie mise en œuvre, présenter les résultats obtenus et leur analyse critique.

Remarques

— les données `bank.csv` contiennent des variables catégorielles qu'il faut encoder sous forme de variables 0-1 (one-hot encoding),

— pour `FMA`, la volumétrie des données est importante.

1. elles sont téléchargeables sur https://github.com/mdeff/fma
2. un descriptif détaillé est sur https://arxiv.org/pdf/1612.01840.pdf

Classification d'images [3]

L'archive image-classification.zip contient un répertoire `data` où se trouvent les données (les images brutes au format jpg ou leurs transformations stockées au format Matlab). Les données à classer sont décrites sur http://www.di.ens.fr/willow/events/cvml2011/materials/practical-classification/. Le type de transformation appliquée aux images y est aussi décrite : de façon simplifiée, des points d'intérêt sont extraits de chaque image. Ces points servent à construire un dictionnaire de mots visuels (via une méthode de clustering). Pour trouver la représentation d'une image, des patch sont extraits puis "projetés" sur ces mots visuels résultant en des histogrammes spatialement organisés qui sont ensuite concaténés

1. Décompresser l'archive ; sous Matlab positionner vous dans le répertoire où les données sont décompressées
2. Lancer le script `setup.m` : si tout se passe bien vous aurez un message de confirmation que l'installation est ok
3. On va dans un premier temps réaliser la classification des images de "motos" (label 1) contre des images d'arrière-plan (label -1) en utilisant un SVM linéaire.
   (a) Compléter le script `maClassifImage.m` pour sélectionner le modèle SVM adéquat
   (b) Visualiser les mots visuels les plus pertinents pour la classification en décommentant la ligne appropriée dans le script. Que constatez-vous ?
   (c) Évaluer les performances du modèle sur les données d'apprentissage (tracer la courbe ROC et mesurer le taux de bonne classification)
   (d) Évaluer les performances du modèle sur les données de test (tracer la courbe ROC et mesurer le taux de bonne classification). Comparer. Déterminer le nombre d'images correctement classées comme "motos" parmi les n (par exemple $n = 24, 36, 48$) premières images selon le score $f(x) = w^\top x + b$ du SVM
   (e) Comment changent les performances avec ou sans normalisation des histogrammes (il suffit de commenter les lignes dans le script) ?
4. On va maintenant étudier l'influence de la représentation utilisée. Plutôt que d'utiliser les histogrammes sur les patches d'images, on projette l'image entière sur les mots visuels ce qui donne un seul histogramme. Pour cela décommenter les lignes appropriées dans le script et apprendre un SVM linéaire. Comment changent les performances ? Quel est l'intérêt de travailler sur des patches plutôt que l'image entière ?
5. BONUS 1 : pour faire notre SVM linéaire nous avons explicitement utiliser ls histogrammes comme des vecteurs car $f(x) = w^\top x + b = \sum_{i=1}^N \alpha_i y_i x_i^\top x + b$. Le produit scalaire $k(x_i, x) = x_i^\top x = \sum_j^d x_i(j)x(j)$ est une mesure de similarité entre x_i et x et donc entre histogrammes. Une mesure de similarité plus pertinente pour les histogrammes est $k(x_i, x) = \sum_j^d \sqrt{x_i(j)x(j)}$. Pour y arriver il suffit de faire un SVM linéaire avec des points $\left(\sqrt{x_i}, y_i\right)_{i=1}^N$ plutôt que $(x_i, y_i)_{i=1}^N$. Modifier votre code matlab et comparer les résultats obtenus avec ceux de la question 3)
6. BONUS 2 : reprendre les questions précédentes pour les catégories "personnes" et "avions"

3. Cet exercice est librement inspiré de http://www.di.ens.fr/willow/events/cvml2011/materials/practical-classification/

Données astro : classification SVM multi-classe

Les données portent sur 3 classes : étoiles (label 1), naines blanches (label 2) et quasars (label 3). Cet énoncé vous propose de réaliser la classification par un SVM multi-classe (autrement, choisir la technique de classification de votre choix). Le nombre de points étant élevé, il vous est conseillé de n'utiliser qu'une partie pour l'apprentissage du SVM linéaire (par souci de rapidité)

1. Afficher le nombre de points par classes
2. On va apprendre un SVM linéaire multi-classe. Pour cela on va utiliser la classification "un contre tous". Etant donné le jeu d'apprentissage $\left\{(x_i, y_i) \in \mathbb{R}^d \times \{1,2,3\}\right\}_{i=1}^N$, son principe est le suivant :
   — Pour chaque classe $\mathcal{C}_k, \quad k = 1,2,3$

     Apprendre un SVM linéaire $f_k(x) = w_k^\top x + b_k$ en prenant la classe $\mathcal{C}_k$ comme celle des points positifs et les autres classes comme des points négatifs c'est-à-dire avec les données $\{(x_i, z_i) \in \mathbb{R}^d \times \{-1,1\}\}_{i=1}^N$ où $z_i = 1$ si $y_i = \mathcal{C}_k$ et $z_i = -1$ autrement

   — Classification d'un point x_ℓ : stratégie du winner takes all

     Pour chaque SVM linéaire calculer la sortie $f_k(x) = w_k^\top x_\ell + b_k, \quad k = 1,2,3$

     Affecter le point x_ℓ à la classe de plus fort score $f_k(x_\ell)$ c'est-à-dire

     $$\hat{y}_\ell = \text{argmax}_{k=1,2,3}\{w_1^\top x_\ell + b_1, w_2^\top x_\ell + b_2, w_3^\top x_\ell + b_3\}$$

   (a) Ecrire une fonction `[matW, vectb] = monsvmmulticlass(X, labels, C)` en utilisant la fonction `monsvmclass` vue en TD. Elle renverra la matrice $W = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}$ et le vecteur $b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$

   (b) Ecrire une fonction `ypred = monsvmmultival(Xval, matW, vectb)` qui renvoie la classe prédite `ypred` pour les points dans `Xval`. On se basera sur la fonction `monsvmval` vue en TD.
3. En utiisant les fonctions écrites essayer de classer les données des trois classes. N'oubliez pas les fondamentaux du machine learning (normalisation, validation croisée, évaluation des performances ...)