BE : Optimisation différentiable 2

Par groupe de 4 étudiants maximum, vous devez rédiger un rapport détaillé dans lequel vous répondrez aux questions du sujet et plus globalement au problème demandé. La qualité de rédaction ainsi que tous les compléments que vous apporterez (réflexions, analyses des réponses, etc...) seront fortement pris en compte. Toutes les fonctions programmées le seront en Python et seront abondamment commentées. Le rendu du projet se fera sous la forme d'un fichier .zip contenant le rapport ainsi que l'ensemble des programmes.

1 Les moindres carrées multi-classes avec régularisation

Nous noterons dans tout cette partie : $Data \in \mathcal{M}_{md}(\mathbb{R})$ une matrice de m données où chaque donnée est donnée par un vecteur ligne de taille d de la matrice $Data$.

On suppose que ces données sont fournis avec leurs catégories (=labels). Notons $C \in \mathbb{N}^*$ le nombre de catégorie des données $Data$.

1.1 Version linéaire

On cherche à apprendre une fonction linéaire :

$$f : \mathbb{R}^{d+1} \to \mathbb{R}^C$$

de la forme :

$$f(x) := x^T W$$

où : $x := (Data_{ligne,i}, 1)^T$ avec $Data_{ligne,i}$ le i ème vecteur ligne de $Data$ et qui renvoie un vecteur ligne de taille C de la forme :

$$b := \begin{cases} b_j = 1 & \text{si le label de } x \text{ est } j \in [\![1, C]\!] \\ 0 & \text{sinon} \end{cases}$$

1. Quel est la taille de la matrice W ? On appellera cette matrice W la **matrice des paramètres**.
2. Justifier que le problème d'apprentissage de la fonction f s'écrit sous forme matricielle :

$$\arg\min_W \frac{1}{2} \|DW - B\|_F^2$$

où D est la matrice par blocs :

$$D := \begin{pmatrix} Data & 1_m \end{pmatrix}$$

et B la matrice dont les coefficients sont de la forme :

$$B_{ij} := \begin{cases} 1 & \text{si le label de } Data_{ligne,i} \text{ est } j \in [\![1, C]\!] \\ 0 & \text{sinon} \end{cases}$$

3. Soit $\rho \in \mathbb{R}_+^*$. Nous allons nous intéresser à la forme dîtes régularisée du problème d'apprentissage :

$$(\mathcal{P}) \quad \arg\min_W \frac{1}{2} \|DW - B\|_F^2 + \frac{\rho}{2} \|W\|_F^2$$

On appelle ρ le **facteur de régularisation**.

(a) Expliquer le ou les intérêt(s) à ajouter le terme $\frac{\rho}{2} \|W\|_F^2$.

(b) Montrer que le problème $(\mathcal{P})$ s'écrit sous la forme d'une forme quadratique matricielle :

$$(\mathcal{P}_1) \quad \arg\min_W \frac{1}{2} \langle W, AW \rangle - \langle C, W \rangle$$

où le produit scalaire est donnée par : $\langle X, Y \rangle := trace\left(X^T Y\right)$ et A et C des matrices à déterminer.

(c) Écrire un algorithme puis un programme sous Python du gradient conjugué adapté à ce produit scalaire et à cette fonction coût. Vous testerez votre code sur :

i. l'ACP de dimension 2 de Iris.csv. Vous tracerez la fonction obtenue grâce à la fonction plt.contour (ou plt.contourf) de Matplotlib.

ii. les données train_data1.npy et train_data2.npy (et leurs labels respectifs).

iii. D'autres données 2D de votre choix.

Vous définirez votre fonction d'apprentissage sous la forme : app_global_MC(train_data,train_labels, nbcat,rho,epsilon) où :

- train_data et train_labels : les données d'entraînements et leurs labels associés,
- nbcat : le nombre de catégorie d'apprentissage,
- rho : le facteur ρ de régularisation,
- epsilon : le facteur d'erreur ϵ de fin de boucle du gradient conjugué.

et qui renvoie en sortie la matrice W d'apprentissage et affichera le nombre d'itération du gradient conjugué. On pourra utiliser les commande suivantes pour évaluer la fonction f sur les données d'entraînements :

```
def f(x,W) :
    y=np.vstack((x,1))
    val=y.T @ W
    return np.argmax(val)
```

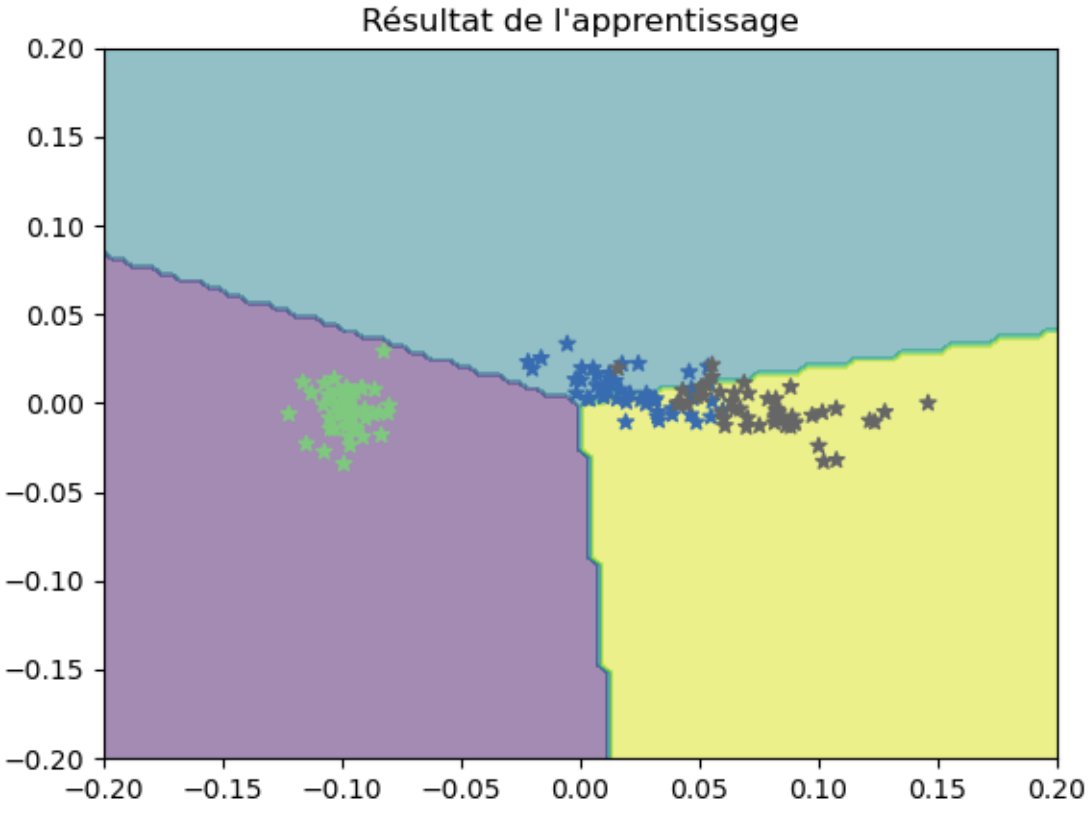


FIGURE 1 – Résultat sur l'ACP de dimension 2 de Iris

4. Pour les données 2D, comparer le résultat obtenue avec un apprentissage avec un réseau de neurone de la forme vu en TD.

5. Appliquer le programme précédent sur les données d'entraînement des bases de données MNIST et Fashion MNIST. Afin de tester les résultats obtenues vous réaliserez une matrice de confusion sur les données tests de chacune de ces deux bases de données de la forme suivante :
   - la matrice de confusion est de taille : $Conf \in \mathcal{M}_{C,C}(\mathbb{N})$ avec C le nombre de catégories.
   - et les coefficients de cette matrice sont donnés par :

$$c_{ij} := \text{"nombre de fois où le label i donne la sortie j"}$$

   Donner le taux de réussite. Commenter et analyser les résultats obtenus.

1.2 Version non-linéaire avec l'astuce des noyaux.

Dans la partie précédente on cherchait à apprendre une fonction linéaire :

$$f : \mathbb{R}^{d+1} \to \mathbb{R}^C$$

de la forme par blocs :

$$f(x) := x^T W = \begin{pmatrix} x^T w_1 & x^T w_2 & \cdots & x^T w_C \end{pmatrix}$$

Nous allons utiliser l'astuce des noyaux sur la forme précédente. On suppose fixée un noyau de type positif $kern$: $\mathbb{R}^{d+1} \times \mathbb{R}^{d+1} \to \mathbb{R}$, et un ensemble de taille $P \in \mathbb{N}^*$:

$$P_K := \{x \in Data\}$$

on sait alors d'après le théorème de Mercer qu'il existe une fonction de redescription $\phi : P_K \to H$ et un produit scalaire $\langle \bullet, \bullet \rangle$ sur H tel que :

$$kern(a, b) = \langle \phi(a), \phi(b) \rangle$$

pour tout $(a, b) \in P_K^2$.

1. Justifier que pour tout élément $w \in H$ et $x := (d, 1)^T$ avec d un vecteur ligne de $Data$ s'écrit sous la forme :

$$\langle \phi(x), w \rangle = \begin{pmatrix} kern(x_1, x) & \cdots & kern(x_P, x) \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_P \end{pmatrix}$$

2. En déduire que l'on peut transformer la version linéaire du problème d'apprentissage de la partie précédente en un problème de la forme :

$$(\mathcal{P}^K) \quad \arg\min_W \frac{1}{2} \left\| D^K W^K - B \right\|_F^2 + \frac{\rho}{2} trace\left(\left(W^K\right)^T K W^K \right)$$

   où :
   - D^K est une matrice de taille $m \times P$ ne dépendant des points de P_K, de $Data$ et de $kern$.
   - W^K : une matrice de paramètres (inconnues donc) de taille $P \times C$.
   - B : la matrice de la partie précédente.
   - K : la matrice de Gram associée à P_K et $kern$.
3. Adapter à la « version noyau » les questions 3 et 4 de la partie précédente. Vous testerez différents noyaux et différents choix de points P_K. En particulier vous définirez votre fonction d'apprentissage sous la forme :

   app_global_MCK(train_data,train_labels, nbcat,rho,epsilon,kern,Pk)

   Dans le cas des bases de données MNIST et Fashion MNIST vous chercherez à atteindre le meilleur taux de réussite. Commenter soigneusement vos résultats dans le rapport. Quelles sont les principales limites de cette approche ?

2 Un algorithme « alternatif » pour les SVM

Les notations de cette partie correspondent au TD 2 sur les SVM. Dans cette partie nous allons décrire un problème de classification « linéaire ». Supposons donnés un ensemble fini E partitionné en deux sous-ensembles disjoints de points de $\mathbb{R}^n$:

$$E_1 = \{u_i \in \mathbb{R}^n \,|\, i \in [\![1,p]\!]\} \quad \text{et} \quad E_2 = \{v_j \in \mathbb{R}^n \,|\, j \in [\![1,q]\!]\}$$

L'idée est de trouver un hyperplan séparateur. On peut se représenter cette situation de la manière suivante : par exemple les points de E_1 vérifient une propriété P alors que les points de E_2 non. On cherche alors à séparer les deux ensembles de points par un hyperplan H. L'espace $\mathbb{R}^n$ sera alors divisé en deux parties disjointes. L'espace situé « au-dessus » de H et l'espace situé « en-dessous » de H. Supposons alors que l'on se donne un nouveau point $x \in \mathbb{R}^n$. Prédire si x vérifie ou non la propriété P revient à regarder où x est situé par rapport à l'hyperplan H .

Nous avons vu en TD que la formulation des SVM revient à résoudre un problème de la forme suivante :

$$(P_2) : \begin{cases} \min \dfrac{1}{2}\|w\|^2 \\ \text{Sous les contraintes : } \begin{cases} w^T u_i - b \geq 1 & \forall\, i \in [\![1,p]\!] \\ -w^T v_j + b \geq 1 & \forall\, j \in [\![1,q]\!] \end{cases} \end{cases}$$

1. Montrer que $J(w,b) := \dfrac{1}{2}\|w\|^2$ est $1-$elliptique et ∇J est $1-$lipschitzienne.
   (a) Démontrer que l'ensemble : $\mathcal{C} =: \left\{(w,b) \in \mathbb{R}^n \times \mathbb{R} \,|\, w^T u_i - b \geq 1, -w^T v_j + b \geq 1, \forall\, i \in [\![1,p]\!], \forall\, j \in [\![1,q]\!]\right\}$ est convexe.
   (b) Déduire des questions précédentes qu'il existe une solution aux problèmes (P_2).
   (c) Montrer que les contraintes s'expriment sous la forme :

$$C \begin{pmatrix} w \\ b \end{pmatrix} + 1_{p+q} \leq 0_{\mathbb{R}^{p+q}}$$

   où C est une matrice à déterminer et 1_{p+q} est le vecteur colonne de $\mathbb{R}^{p+q}$ contenant uniquement des 1.
2. Soit $z := (z_1, z_2, \cdots, z_{p+q})^T$ tel que

$$C \begin{pmatrix} w \\ b \end{pmatrix} + z + 1_{p+q} = 0_{\mathbb{R}^{p+q}}$$

   On considère ce que l'on nomme le **lagrangien augmenté** suivant :

$$\mathcal{L}_\rho(w,z,\lambda) = \frac{1}{2}\|w\|^2 + \lambda^T \left(C \begin{pmatrix} w \\ b \end{pmatrix} + z + 1_{p+q} \right) + \frac{\rho}{2} \left\| C \begin{pmatrix} w \\ b \end{pmatrix} + z + 1_{p+q} \right\|_2^2$$

   avec $\lambda \in \mathbb{R}^{p+q}$ est le multiplicateur de lagrange et $\rho > 0$ est le paramètre dit de pénalité quadratique des contraintes.

   Nous allons utiliser pour la résolution numérique du problème précédent le schéma itératif d'ADMM (*Alternating Direction Method of Multipliers*) défini par :

$$\left\{ \begin{array}{c} w^{k+1} = \arg\min_w \mathcal{L}_\rho\left(w, z^k, \lambda^k\right) \\ z^{k+1} = \arg\min_z \mathcal{L}_\rho\left(w^{k+1}, z, \lambda^k\right) \\ \lambda^{k+1} = \lambda^k - \rho\left(Cw^{k+1} + z^{k+1} + 1_{p+q}\right) \end{array} \right\| \quad (P)$$

   (a) En posant $x = (w,b)^T$ montrer que :

$$\mathcal{L}_\rho(x,z,\lambda) = \frac{1}{2}\|Ax\|^2 + \lambda^T(Cx + z + 1_{p+q}) + \frac{\rho}{2}\|Cx + z + 1_{p+q}\|_2^2$$

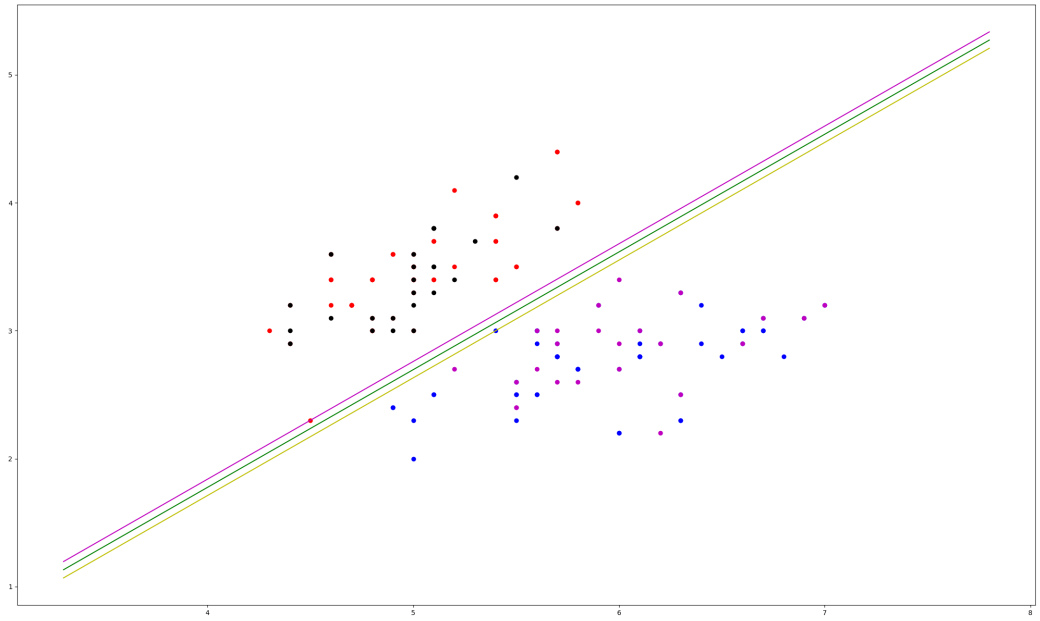
   avec A une matrice à déterminer.
   (b) Montrer que

$$\left\{ \begin{array}{c} \frac{\partial}{\partial x}\mathcal{L}_\rho(x,z,\lambda) = \left(A + \rho C^T C\right)x + C^T\left(\rho(z+d) + \lambda\right) \\ \frac{\partial}{\partial z}\mathcal{L}_\rho(x,z,\lambda) = \lambda + \rho(z + Cx + d) \end{array} \right.$$

   (c) Écrire une fonction python Admm() permettant d'implémenter l'algorithme (P).

Pour tester programme, on pourra utiliser la base de données iris .

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
# Chargement des données
iris = datasets.load_iris()
#on ne prend que les deux premières colonnes d'iris.
X, y = iris.data[:, :2], iris.target
# On conserve 50% du jeu de données pour l'évaluation
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5)# on prend que deux gategories
u_train=X_train[y_train==0,:]
v_train=X_train[y_train==1,:]
u_test=X_test[y_test==0,:]
v_test=X_test[y_test==1,:]
```



3. Utiliser l'algorithme ADMM pour réaliser un programme Python de reconnaissance des chiffres manuscrits à partir de la base de données MNIST et (puis) d'habits avec Fashion MNIST. Comparer les résultats obtenus avec les résultats de la première partie.
4. Donner pour chacun des cas la matrice de confusion sur les données tests de chacune de ces deux bases de données MNIST de la forme suivante :
   - la matrice de confusion est de taille : $Conf \in \mathcal{M}_{C,C}(\mathbb{N})$ avec C le nombre de catégories (10 pour MNIST).
   - et les coefficients de cette matrice sont donnés par :

   $$c_{ij} := \text{"nombre de fois où le label i donne la sortie j"}$$

   Donner le taux de réussite. Commenter et analyser les résultats obtenus.
5. (Bonus) : Appliquer l'algorithme ADMM à la formulation avec noyau de la SVM. En choisissant convenablement le noyau et les paramètres en jeux trouver le plus haut taux de reconnaissance possible sur les données tests pour les deux bases de données MNIST.