TP 6 SVM

Reprenons l'exemple du cours : $\mathbf{X} = \begin{bmatrix} 0 & 0 \\ 2 & 2 \\ 2 & 0 \\ 3 & 0 \end{bmatrix}$ et $y = \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$

I. Plan séparateur

Utiliser la fonction aff_donnees($\mathbf{X}, y, bornex, borney, s$) qui affiche les données sur une figure en mettant un symbole différent pour les exemples positifs et négatifs. $bornex$, $borney$ représentent les limites des axes que l'on pourra fixer à min-1 et max+1. s représente la taille des symboles (utile dans la dernière partie du TP) que l'on pourra fixer à 50.

```
def aff_donnees(X,y,bornex,borney,s):
    plt.scatter(X[:, 0], X[:, 1], c=y, s=s, cmap='winter');
    plt.xlim(bornex);
plt.ylim(borney);
```

Ecrire une fonction affichePlan($\mathbf{w}, b, bornex$) qui affiche un hyperplan dans le plan 2D en passant en argument les paramètres de l'hyperplan $\boldsymbol{w}$ et b ainsi que $bornex$. On pourra pour cela faire varier x dans l'intervalle défini par $bornex$ et calculer les y correspondants. Afficher l'hyperplan de paramètre $\boldsymbol{w}^T = [1{,}0.1]$ et $b = -1$.

Questions

- S'agit-il d'un hyperplan séparateur ?

II. SVM linéaire dans le primal

On a vu en cours que le problème des SVM peut être mis sous la forme d'un problème quadratique dont la solution est obtenue avec la librairie cvxopt. Compléter la fonction ResoudPrimal(**X**,y) qui renvoie les paramètres $\boldsymbol{w}$ et b de l'hyperplan séparateur.

Rq : la fonction sol = cvxopt.solvers.qp(P, q, G, h) résout le problème :

$$\min_x 0.5x^T P x + q^T x \qquad sc \quad Gx \leq h$$

P, q, G, h doivent être des matrices de décimaux de type cvxopt.matrix. La conversion est réalisée avec P= cvxopt.matrix(P). La solution au problème est trouvée avec x=sol['x'].

Compléter pour cela le code suivant :

```
def Resoud_primal(X,y):
N= ???
N= ???
q = ???
P1=np.concatenate((np.zeros((1,1)),np.zeros((1,n))),axis=1)
P2=np.concatenate((np.zeros((n,1)),np.eye(n)),axis=1)
P=np.concatenate((P1,P2),axis=0)
P=cvxopt.matrix(P)
for i in range(N):
    g=np.concatenate((np.reshape(-y[i],(1,1)), np.reshape(-y[i]*X[i][:],(1,2))),axis=1)
    if i==0:
        G=g
    else:
```

```
    G=np.concatenate((G, g), axis=0)
G=cvxopt.matrix(G+0.)
h = ???
sol = ???
x=???
b=???
w=???
return w,b
```

Pour concaténer des matrices A et B horizontalement, on fera :
`np.concatenate((A,B),axis=1)`
pour les concatener verticalement,
`np.concatenate((A,B),axis=0)`
Pour remplir une matrice de taille *dy*x*dx* avec des uns ou des zéros, on fera
`np.zeros((dy,dx)) ou np.ones((dy,dx))`
Une matrice identité de taille *dx*x*dx* sera obtenue avec
`np.eye(dx)`
Vérifier votre programme en traçant cet hyperplan.

Questions

- Est-ce que l'hyperplan obtenu vous parait correct ?
- Que se passe-t-il si on ajoute le point $\mathbf{x} = \begin{bmatrix} 2.1 \\ 2.5 \end{bmatrix}$ d'étiquette 1 ? Vérifier en faisant tourner le programme.
- Même question avec le point $\mathbf{x} = \begin{bmatrix} 1.5 \\ 2.5 \end{bmatrix}$

III. SVM à marge souple

On va utiliser l'apprentissage SVM de python qui autorise les marges souples.

```
model = svm.SVC(kernel='linear', C=???)
model.fit(X, y)
w = model.coef_[0]
b = model.intercept_[0]
```

Afficher l'hyperplan obtenu avec la fonction aff_plan() précédente puis avec la fonction aff_frontiere() donnée en annexe. Que réalise la fonction aff_frontiere() ?

Questions

- Que représente la variable C
- Tester le programme sur les données initiales. Conclusion ?
- Ajouter le point $\mathbf{x} = \begin{bmatrix} 2.1 \\ 2.5 \end{bmatrix}$ d'étiquette 1 et tester plusieurs valeurs de C. Conclusion ?
- Ajouter le point $\mathbf{x} = \begin{bmatrix} 1.5 \\ 2.5 \end{bmatrix}$ d'étiquette 1 et tester plusieurs valeurs de C. Conclusion ?

IV. SVM avec Kernel

Dans cette partie, on travaillera sur les données «TP6.npz » :

```
f = np.load('TP6.npz',allow_pickle=True)
X=f['arr_0']
```

```
y=f['arr_1']
```

Apprendre directement un SVM avec un kernel linéaire et afficher fonction de décision avec aff_frontiere(X,y,bornex,borney,model).

Apprendre des SVM avec des noyaux linéaires gaussien, polynomial et linéaire. Que remarquez-vous sur les nouvelles fonctions de décisions ?

Questions

- Peut-on apprendre des frontières non linéaires avec un noyau linéaire ?
- Que donne un SVM avec noyau rbf sur les données linéairement séparable (données de départ) ? Est-il bien adapté ?

Annexe

V. Annexe

```
def aff_frontiere(X,y,bornex,borney,model):
    aff_donnees(X,y,bornex,borney,50)

    xx, yy = np.meshgrid(np.linspace(bornex[0], bornex[1],50), np.linspace(borney[0], borney[1],50))

    xy = 
np.concatenate((np.reshape(xx,(xx.shape[0]*xx.shape[1],1)),np.reshape(yy,(yy.shape[0]*yy.shape[1],
1))),axis=1)

    P = model.predict(xy)

    aff_donnees(xy,P,bornex,borney,1)
```