IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

UVF3B403 MS IABDA

TP sur les SVM

Yannis Haralambous (Télécom Bretagne)

Dans ce TP nous allons d'abord décortiquer un exemple élémentaire de SVM et puis voir quelques exemples d'utilisation de SVM sous Python. Les packages utilisés seront `scikit-learn` (et ses sous-packages `svm` et `datasets`), `numpy` et `matplotlib`.

1 Un SVM élémentaire

Prenons dans le plan euclidien $\mathbb{R}^2$ les individus $(1,1)$, $(1,5)$, $(5,1)$ et $(5,5)$, associés aux classes $-1,1,1,1$.

1) Calculer manuellement le vecteur (a,b) ainsi que la valeur de c de la droite qui sera le séparateur optimal.

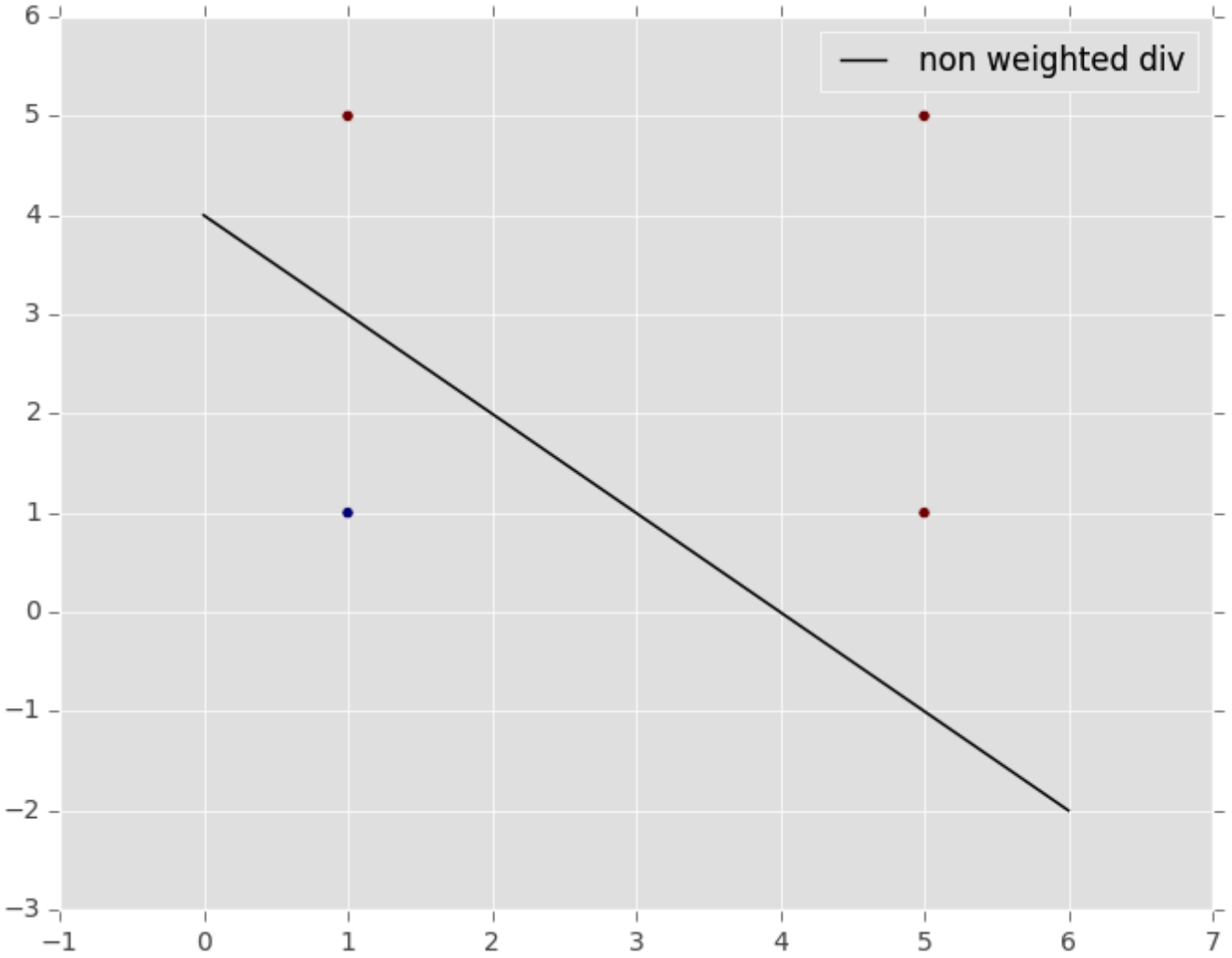
2) Implémenter (bien comprendre le code) :

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
style.use("ggplot")
from sklearn import svm
X = np.array([[1, 1], [1, 5], [5, 1], [5, 5]])
y = [-1,1,1,1]
clf = svm.SVC(kernel='linear')
clf.fit(X, y)
print clf.support_vectors_
print clf.n_support_

w = clf.coef_[0]
print(w)
a = -w[0] / w[1]
xx = np.linspace(0,6)
yy = a * xx - clf.intercept_[0] / w[1]

plt.plot(xx, yy, 'k-', label="non weighted div")
plt.scatter(X[:, 0], X[:, 1], c = y)
plt.legend()
plt.show()
```

pour obtenir



Que signifient les différents résultats obtenus :

```
[[ 1.  1.]
 [ 1.  5.]
 [ 5.  1.]]
[1 2]
[ 0.49975586  0.49975586]
```

2 Quelques exemples d'applications

On a le choix entre quatre noyaux (option `kernel` de `SVC`) :

1. linéaire `linear` $k(x, x') = \langle x, x' \rangle$;
2. polynomial `pol` $k(x, x') = (\gamma \cdot \langle x, x' \rangle + r)^d$;
3. radial `rbf` $k(x, x') = e^{-\gamma \|x - x'\|}$ (par défaut) ;
4. sigmoïde `sigmoid` $k(x, x') = \text{th}(\gamma \cdot \langle x, x' \rangle)$.

Les paramètres γ, d et r s'écrivent `gamma`, `degree` et `coef0` resp. Le paramètre de coût C (vu en cours) s'écrit `cost`.

2.1 Iris

Nous allons commencer par les illustres données « iris » de Fischer : la taille en centimètre des pétales et autres parties que je ne saurais nommer de certaines fleurs. Pour chaque individu on a 4 infos numériques

et la classe. Les classes sont *Iris setosa*, *Iris versicolor* et *Iris virginica*. On a 150 individus et les classes sont équidistribuées.

Faire

```
from sklearn import svm
from sklearn import datasets
clf = svm.SVC()
iris = datasets.load_iris()
X, y = iris.data, iris.target
```

À partir des données `iris` créer un ensemble d'entraînement `train_X,train_y` avec 100 individus tirés au hasard, et un ensemble de test `test_X,test_y` avec ce qui reste. *Tuyau : utiliser la fonction `shuffle` du module `random`.*

Lancer le SVM en écrivant

```
clf.fit(train_X,train_y)
```

Pour faire des prédictions, utiliser

```
clf.predict(test_X)
```

Calculer, en utilisant des compréhensions de listes Python, la précision et le rappel de chaque classe.

En utilisant la classe `KFold` du package `sklearn.cross_validation`, écrire le code d'une 10-validation croisée et obtenir la précision et le rappel moyens pour chaque classe.

Faire varier les types de noyau et les paramètres correspondants pour voir de quelle manière cela varie.

2.2 SPAM

Refaire toutes les opérations en utilisant l'ensemble de données « SPAM » du labo Hewlett-Packard (`https://archive.ics.uci.edu/ml/datasets/Spambase`). Les classes sont *spam* et *nonspam*. Pour chaque individu on a 57 données numériques et la classe. On a 4 061 individus et les classes sont distribuées de la manière suivante : 39,4% de spam et 60,59% de nonspam.

Voir

`https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.DOCUMENTATION` pour la description des données.

Faire varier les noyaux et les paramètres et comparer les résultats en fonction de l'importance de la précision et du rappel.