TD Machines à vecteurs de support

1. Introduction

Le prochain type d'apprentissage supervisé que nous allons aborder est appelé *SVM à noyau*. Le sigle SVM signifie *Support Vector Machines*, soit littéralement *machines à vecteurs de support*, mais que l'on traduit généralement par *Séparateur à Vaste Marge* pour conserver les initiales. Les SVM s'appliquent à la classification comme à la régression. Même si les notions mathématiques qui sous-tendent les SVM à noyau vont bien au-delà de l'objet de ce TD, nous allons tout de même essayer de comprendre dans ce qui suit l'idée générale de cette méthode. Les matheux pourront toujours consulter le chapitre 12 de l'ouvrage de référence *Elements of Stastical Learning* de Hastie, Tibshirani et Friedman, disponible gratuitement à l'adresse https://web.stanford.edu/~hastie/Papers/ESLII.pdf

2. Classification SVM linéaire

Pour expliquer l'idée fondamentale des SVM, rien de tel que des figures. La figure 1 qui suit présente un jeu de données (C_1 = {(-2,-2),(-1,-1)},C_2={(1,1),(2,2)}) composé de deux classes C_1 et C_2. Il est clair que les deux classes peuvent être facilement séparées par une ligne droite (elles sont *linéairement séparables*). La Figure 1 présente les frontières de décision de trois classificateurs linéaires possibles : le modèle dont la frontière de décision est représentée par une ligne à tirets est si mauvais qu'il ne sépare pas correctement les classes.

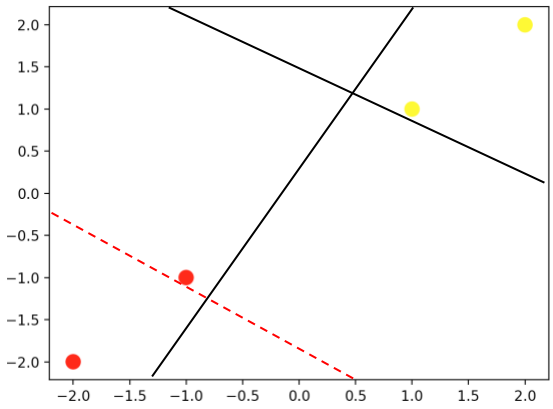

Figure 1 : Classification à vaste/large marge

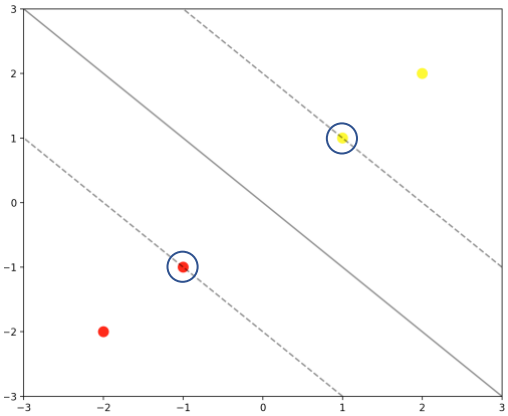

Figure 2 : Classification à large marge

Les deux autres modèles fonctionnent parfaitement sur le jeu d'entraînement, mais leurs frontières de décision sont si proches des observations que ces modèles ne donneront probablement pas d'aussi bons résultats sur de nouvelles observations. Au contraire, la ligne continue de la Figure 2 représente la frontière de décision d'un classificateur SVM : cette ligne ne se contente pas de séparer les deux classes, mais elle reste aussi distante que possible des observations d'entraînement les plus proches. Un classificateur SVM ajuste le chemin le plus large possible (matérialisé par les deux lignes pointillées parallèles) entre les classes. C'est ce qu'on appelle une ***classification à large/vaste marge.*** La frontière de décision est entièrement déterminée par les observations situées sur les bords du chemin (points (-1,-1) et (1,1)). Ces données sont appelées ***vecteurs de support.***

Exercice 1 : *Les vecteurs supports sont supposés influencer le calcul de l'hyperplan. Modifier le script* **ex1_svm.py** *et plus particulièrement le fichier* `ex1data.csv` *afin d'enlever des données d'entraînement les points qui correspondent aux vecteurs supports. Observer le résultat sur le calcul de l'hyperplan.*

Remarque : Les SVM sont sensibles aux différences d'échelle des variables, comme vous pouvez le voir sur la Figure 3 : sur le graphique de gauche, l'échelle verticale est beaucoup plus grande que l'échelle horizontale, c'est pourquoi le chemin le plus large est proche de l'horizontale. Après normalisation (en utilisant par exemple la fonction `StandardScaler` de Scikit-Learn), la frontière de décision paraît bien meilleure (sur le graphique de droite).

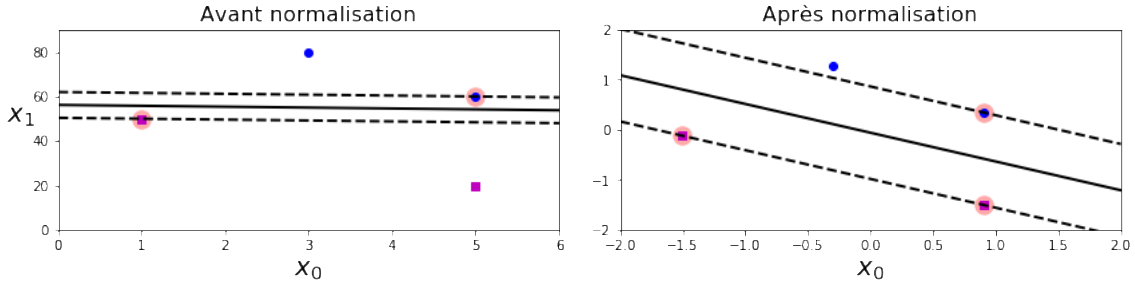

Figure 3 : sensibilité aux échelles des variables

Si nous imposons que toutes les observations soient en dehors du chemin et du bon côté, nous effectuons une classification à ***marge rigide*** (en anglais, ***hard margin*** classification). Une classification de ce type présente deux problèmes principaux :

- Elle ne fonctionne que si les données sont linéairement séparables ;
- Elle est relativement sensible aux données aberrantes.

La Figure 4 présente le jeu de données Iris auquel on a ajouté simplement une donnée aberrante et il est impossible de trouver une marge rigide.

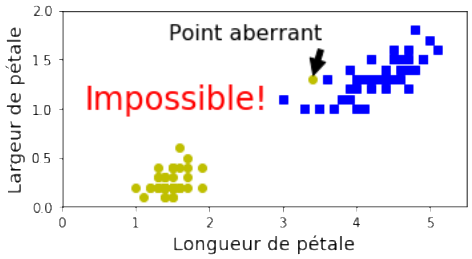

Figure 4 : La marge rigide est très sensible aux données aberrantes

La Figure 5 présente une classification sur le jeu de données Iris dans le cas où les données ne contiennent pas de donnée aberrante (à gauche) et le cas où le jeu de donnée contient une donnée aberrante. On peut s'apercevoir que sur la Figure de droite la frontière de décision est très différente de celle obtenue sans donnée aberrante et qu'elle ne se généralisera probablement pas bien.

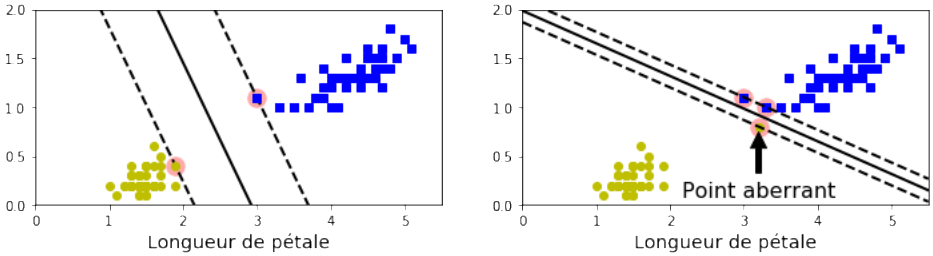

Figure 5 : marge rigide et donnée aberrante

Pour éviter ces problèmes, il est préférable d'utiliser un modèle « plus accommodant » : l'objectif est de trouver un bon équilibre entre conserver un chemin aussi large que possible et limiter les empiètements de marge (c'est-à-dire le nombre d'observations se retrouvant à l'intérieur du chemin, voire même du mauvais côté). C'est ce qu'on appelle une *classification à marge souple* (en anglais, *soft margin*).

Au niveau des classes SVM de Scikit-Learn, vous pouvez contrôler cet équilibre à l'aide de l'hyperparamètre *C* : plus la valeur de *C* est petite, plus le chemin sera large, mais plus vous aurez d'empiètements de marge.

Exercice 2 :

1. *Modifier le script* **`ex2_svm.py`** *afin de prendre en compte le jeu de données* **`ex2data.csv`**. *Observer le comportement par défaut du classifieur* **`LinearSVC()`** *de Scikit-learn ;*
2. *Noter l'influence de l'hyperparamètre C de* **`LinearSVC()`** *pour différentes valeurs de C ;*
3. *Existe-t-il une valeur de C permettant d'obtenir une classification sans erreur ?*

3. Cas non linéairement séparable

3.1 Introduction

Commençons par exemple sous forme d'exercice.

Exercice 3 : Exemple du XOR

Modifier le script **`ex3_xor.py`** afin de répondre aux questions qui suivent :

1. Générer l'ensemble d'entrainement suivant :

   X = (0,0 y = (1
   1,0 0
   0,1 0
   1,1) 1)

2. Afficher cet ensemble d'entraînement avec les labels en couleurs.
3. Entraîner un SVM linéaire (`from sklearn.svm import LinearSVC`)
4. Que constatez-vous ? Quelle explication simple pouvez-vous proposer ?

3.2 Classification SVM non linéaire

Abordons maintenant le problème des données non linéairement séparables. Pour rappel, des données sont non linéairement séparables quand il n'existe pas d'hyperplan capable de séparer correctement les deux catégories. Ce qui, d'ailleurs, est très souvent le cas en pratique. Pour contourner le problème, l'idée est la suivante : il est impossible de séparer linéairement les données dans notre espace vectoriel ? Qu'importe ! Essayons dans un autre espace. De façon générale, il est courant de ne pas pouvoir séparer les données parce que l'espace est de trop petite dimension[1]. Si l'on arrivait à "transposer" les données dans un espace de plus grande dimension, on arriverait, peut-être, à trouver un hyperplan séparateur.

Commençons par un exemple très simple. Supposons que nous souhaitions trouver un hyperplan pour discriminer les points de la figure qui suit.

Figure 6 : Ensemble de points non linéairement séparables

Cependant, si nous "envoyons" les données d'entrées (de $\mathbb{R}$) dans un espace en 2D ($\mathbb{R}^2$) avec la fonction $(y_1, y_2) = \phi(x) = (x, x^2)$. Les points bleus et les points rouges peuvent être séparés par une droite dans le nouvel espace.

[1] https://en.wikipedia.org/wiki/Cover's_theorem

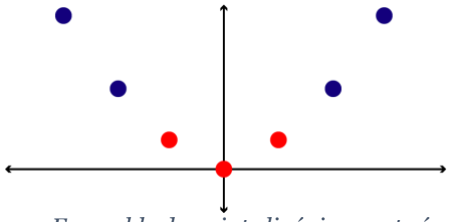

Figure 7 : Ensemble de points linéairement séparable

Voyons un exemple un peu plus compliqué. Soit l'ensemble de point (dans $\mathbb{R}^2$) de la figure qui suit :

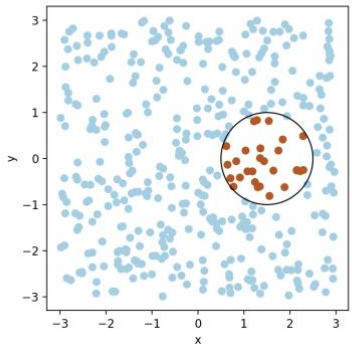

Figure 8 : Ensemble de points non linéairement séparables

Cette fois en considérant la fonction $(x', y', z') = \phi(x, y) = (x, y, x^2 + y^2)$ qui projette les données dans un espace 3D ($\mathbb{R}^3$). Les points bleus et rouges peuvent être séparés par un plan dans le nouvel espace :

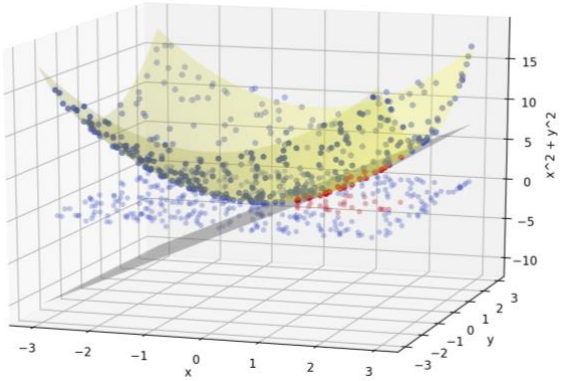

Figure 9 : Les données sont séparables par un hyperplan

3.2.1 Quand faire la projection

Dans l'exemple ci-dessus, on peut s'imaginer qu'il faut d'abord projeter les points avant d'effectuer la classification. En réalité, les SVM vont faire le travail pour nous. C'est toute l'astuce, extrêmement efficace, du ***noyau*** pour faire ces projections.

Avant d'aborder la notion de noyau, faisons le point sur ce que nous avons vu jusqu'à présent :

- Pour des données linéairement séparables linéairement, les SVM fonctionnent étonnamment bien.
- Pour les données qui sont presque linéairement séparables, les SVM peuvent encore fonctionner assez bien en utilisant la bonne valeur de C.

- Pour les données qui ne sont pas linéairement séparables, nous pouvons projeter les données dans un espace où elles sont parfaitement/presque linéairement séparables, ce qui permet de revenir au cas 1 ou 2.

Évidemment, il n'est pas envisageable de calculer l'image dans un espace plus grand de tous les points d'origines. En effet, ce calcul serait beaucoup trop coûteux. L'astuce du noyau nous permet d'éviter ce calcul fastidieux. En fonction du problème à résoudre, il existe plusieurs noyaux. Il est même possible d'imaginer de nouveau noyaux.

3.2.2 Noyau Polynomial

L'ajout de variables polynomiales est simple à réaliser et peut donner de bons résultats avec toutes sortes d'algorithmes d'apprentissage automatique (et pas seulement les SVM), mais avec un degré polynomial faible on ne peut pas tirer parti de jeux de données très complexes, tandis qu'avec un degré polynomial élevé, on obtient un très grand nombre de variables, ce qui ralentit trop le traitement.

Heureusement, lorsqu'on utilise des SVM, il est possible d'appliquer une technique mathématique presque miraculeuse appelée ***l'astuce du noyau*** (kernel trick). Elle permet d'obtenir le même résultat que si vous aviez de nombreuses variables polynomiales, même de degré très élevé, sans avoir à les ajouter effectivement. Il n'y a donc pas d'explosion combinatoire du nombre de variables. Cette astuce est implémentée par la classe `SVC`.

Exercice 4 : Exemple du XOR le retour

Modifier le script **`ex4_xor.py`** afin de répondre aux questions qui suivent :

1. Entraîner un SVM non-linéaire avec un noyau polynomial de degré 3 (fonction `SVC` de scikit-Learn).
2. Essayer différentes valeurs pour les paramètres `degree` et `C` et essayer d'en déduire leurs influences.

3.2.3 Noyau radial Gaussien

Une autre technique de résolution des problèmes non linéairement séparables consiste à ajouter des variables calculées à l'aide d'une *fonction de similarité* qui mesure la ressemblance entre chaque observation et un *point de repère* (en anglais, *landmark*) particulier. Prenons par exemple le jeu de données unidimensionnel représenté sur la Figure 10.

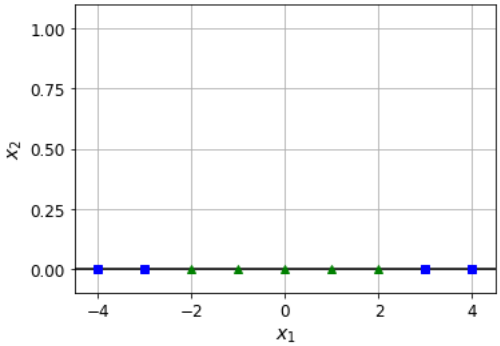

Figure 10 : données non linéairement séparables

Nous allons choisir deux points de repère sur l'axe X_1, $x_1 = -2$ et $x_2 = 1$. Choisissons ensuite comme fonction de similarité une *fonction de base radiale* (en Anglais, *Radial Basis Function* ou *RBF*) gaussienne où $\gamma = 0.3$:

$$\phi(\mathbf{x}, l) = \exp(-\gamma||\mathbf{x} - l||^2) = \exp(-0.3||\mathbf{x} - l||^2)$$

C'est une courbe en cloche variant de zéro (très loin du point de repère) jusqu'à 1 (au point de repère).

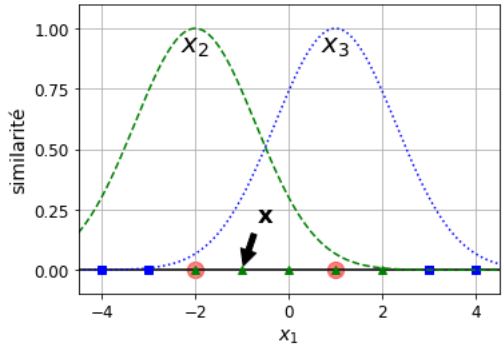

Figure 11 : Variables de similarité avec une RBF Gaussienne

Maintenant, nous sommes prêts à calculer les nouvelles variables. Considérons par exemple l'observation $x_1 = -1$: elle est située à la distance 1 du premier point de repère, et 2 du second point de repère. Par conséquent, les nouvelles variables sont :

$$x_2 = \exp(-0.3 \times 1^2) = 0.74,$$

et

$$x_3 = \exp(-0.3 \times 2^2) = 0.30.$$

La figure 12 présente le jeu de données transformé (en supprimant les variables d'origine). Comme vous pouvez le constater, il est maintenant linéairement séparable.

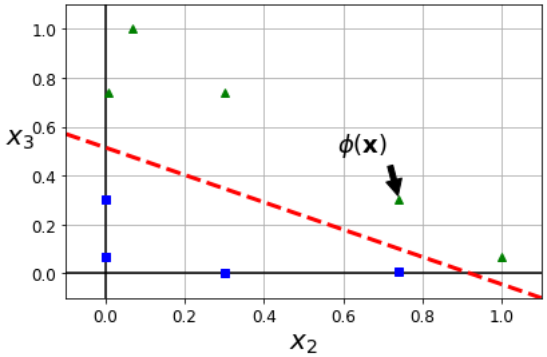

Figure 12 : jeu de données linéairement séparables

Comment sélectionner les points de repère ? La méthode la plus simple consiste à créer un point de repère à l'emplacement de chacune des observations du jeu de données. Ceci crée un très grand nombre de dimension et accroît les chances que le jeu de données transformé soit linéairement séparable. L'inconvénient, c'est qu'un jeu d'entraînement de m observations et de n variables est transformé en un jeu d'entrainement de m observations et m variables (en supposant que vous abandonniez les variables d'origine). Si le jeu de données d'entrainement comporte beaucoup d'observations, nous aboutissons à un nombre tout aussi important de variables.

Dans ce cas, nous allons, également, pouvoir utiliser l'astuce du noyau. Elle permet d'obtenir un résultat semblable à ce qu'on aurait en ajoutant toutes les variables de similarité, mais sans les ajouter effectivement.

Exercice 5 : Noyau Radial Gaussien

1. Modifier le script `ex5_RBF.py` afin de comprendre l'influence des hyper paramètres γ et C.
2. Reprendre le script `ex4_xor.py` afin d'appliquer un noyau Radial Gaussien

3.2.4 Les autres noyaux

Il existe d'autres noyaux, mais qui sont utilisés plus rarement : par exemple certains noyaux sont adaptés à des structures de données spécifiques : on utilise parfois des noyaux de chaînes pour la classification de documents textuels ou de séquence d'ADN (par exemple un noyau de sous-chaînes ou un noyau basé sur la *distance de Levenshtein*).

Comment faire son choix parmi les nombreux noyaux proposés. En règle générale, il faut commencer par essayer le noyau linéaire en particulier si le jeu d'entraînement comporte beaucoup d'observations ou s'il a beaucoup de variables. Si le jeu d'entraînement n'est pas trop grand, essayez également un noyau radial gaussien (TBF) : il donne de bons résultats dans la plupart des cas.

Exercice 6 : Classification de vins
Dans cet exercice, nous allons utiliser les données concernant les caractéristiques physico-chimiques de vins rouges portugais disponibles sur l'archive UCI https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/. Il s'agit ici de prédire le score (entre 3 et 9) donné par des experts aux différents vins.

Questions :

1. La première étape est de transformer notre problème en un problème de classification. Après avoir chargé les données, nous allons créer deux classes : les vins de bonne qualité (score supérieur à 6) et les autres… Il s'agira de prédire en fonction des caractéristiques physico-chimiques du vin à qu'elle classe il appartient :

```
import numpy as np
# charger les donnees
import pandas as pd
data = pd.read_csv('winequality-red.csv', sep=';')
print(data.head(20))

# creer la matrice de donnees
X = data.drop('quality',axis=1).copy()

# creer le vecteur d'etiquettes
y = data['quality'].values
y = y.flatten()
print(y)

# transformer en un probleme de classification binaire
y_class = np.where(y<6, 0, 1)
```

2. Ajouter au script précédent le code python (fonction `model_selection.train_test_split`[2] de scikit-learn) qui permet de découper nos données en un jeu d'entraînement (`X_train`, `y_train`) et un jeu de test (`X_test`, `y_test`).
3. Il faut normaliser http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html les variables, c'est-à-dire les centrer (ramener leur moyenne à 0) et les réduire (ramener leur écart-type à 1), afin qu'elles se placent toutes à peu près sur la même échelle.
4. Tout est prêt pour pouvoir entraîner le SVM à noyau. Ajouter le code python permettant d'entraîner un SVM avec un noyau radial gaussien de paramètre gamma=0.01.
5. A présent pour comprendre comment le SVM se comporte sur le jeu de test, nous allons regarder la courbe ROC. Ajouter le code python qui suit :

[2] http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html#sklearn.model_selection.train_test_split

```
# predire sur le jeu de test
y_test_pred = classifier.decision_function(X_test_std)

# construire la courbe ROC
from sklearn import metrics
fpr, tpr, thr = metrics.roc_curve(y_test, y_test_pred)

# calculer l'aire sous la courbe ROC
auc = metrics.auc(fpr, tpr)

# creer une figure
from matplotlib import pyplot as plt
fig = plt.figure(figsize=(6, 6))

# afficher la courbe ROC
plt.plot(fpr, tpr, '-', lw=2, label='gamma=0.01, AUC=%.2f' % auc)

# donner un titre aux axes et au graphique
plt.xlabel('Taux de faux positifs', fontsize=16)
plt.ylabel('Taux de vrais positifs', fontsize=16)
plt.title('Courbe ROC SVM', fontsize=16)

# afficher la legende
plt.legend(loc="lower right", fontsize=14)

# afficher l'image
plt.show()
```

Vous devez obtenir le graphique suivant :

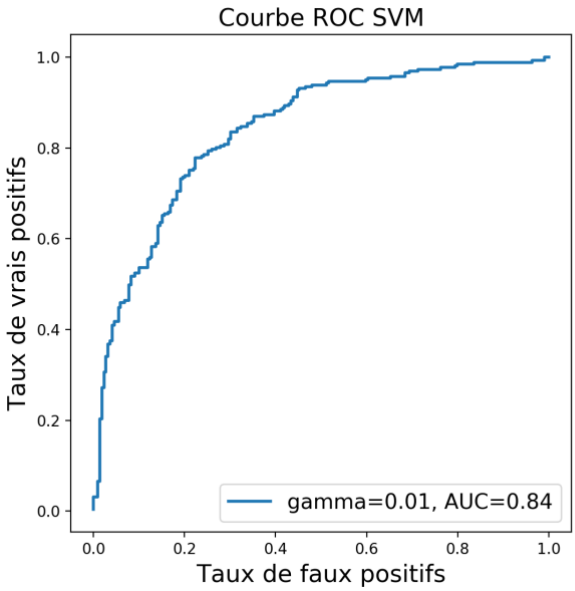

Figure 13 : Courbe ROC - Classification de vins

6. Commenter cette courbe ROC. Relancer des tests en faisant varier les paramètres *gamma* et *C* dans votre SVM.