Big Data
Travaux pratiques
Feuille 2 (Classification non-linéaire)

Dans ce TP, nous allons tenter de retrouver des classes qui ne peuvent pas être correctement retrouver par un classifieur linéaire. On va donc, selon le principe des SVM, transformer les données de l'espace image vers un espace des caractéristiques ("feature space") par une transformation non-linéaire. Nous allons essayer un cas de transformation explicite et un cas de transformation implicite par un noyau Gaussien.

Comme transformation explicite, nous considérons la transformation polynomiale

$$x = (x_1, x_2) \to \Phi(x) \triangleq (x_1^i x_2^j)_{1 \leq i+j \leq \texttt{dPoly}}\,. \tag{1}$$

Exercice 1. Dans cet exercice, nous allons utiliser les svm en tant que machine non-linéaires. Nous allons toujours considérer le cas où les données X_i sont uniforméments distribuées dans le carré $[-1,1]^2$. On va cependant considérer un cas non-linéaire en supposant que la classe de x est donnée par (c'est une forme de damier)

$$y = \begin{cases} 1 & \text{, si } x_1 \in \left([-1,-\frac{1}{2}] \cup [0,\frac{1}{2}]\right) \text{ et } x_2 \in \left([-1,-\frac{1}{2}] \cup [0,\frac{1}{2}]\right) \\ 1 & \text{, si } x_1 \in \left([-\frac{1}{2},0] \cup [\frac{1}{2},1]\right) \text{ et } x_2 \in \left([-\frac{1}{2},0] \cup [\frac{1}{2},1]\right) \\ -1 & \text{, sinon,} \end{cases}$$

où $x = (x_1, x_2)$.

On prend `nbApp` $= 300$ et `nbTest` $= 1000$.

(1) Ecrire une fonction qui construit un échantillon d'apprentissage de taille `nbApp` et un échantillon de test de taille `nbTest`

(2) Ecrire une fonction qui, pour un degré dPoly donné, applique le calcul de feature et transforme l'échantillon $(x_i, y_i)_{i=1..nbApp}$ en un échantillon $(\phi(x_i), y_i)_{1=1..nbApp}$.

(3) Utiliser la bibliothèque *svm* de *scikit-learn* pour calculer, pour une valeur de C donnée, la SVM utilisant le noyau linéaire pour classifier l'échantillon transformée $(\phi(x_i), y_i)_{1=1..nbApp}$.

(4) Ecrire un fonction permettant d'afficher l'échantillon d'apprentissage (avec une couleur dépendant de la valeur de y) et l'échantillon de test (en utilisant un signe permettant de distinguer les examples mals classés.)

(5) Tracer, pour une valeur $dPOly$, les courbes d'apprentissage donnant l'erreur calculée sur l'echantillon d'apprentissage (le risque empirique) et l'erreur calculée sur l'échantillon de test (le risque) en fonction de C.

(6) Observer l'évolution des performances de la machine en fonction de $dPOly = 1..8$. Quelle est la meilleure performance atteinte sur l'ensemble de test. Donner la valeur de `dPoly`, et `C` correspondante.

(7) Copier-coller puis modifier vos programmes pour pouvoir refaire l'expérience pour des SVM utilisant le noyaux Gaussiens. Donner la valeur de γ et de C donnant les meilleurs résultats et comparer avec le noyau polynomiale.

Exercice 2. Dans l'exercice précédent, vous avez du trouver que le noyau Gaussien est meilleur pour prédire la classe de x. Le choix du noyau dépend en fait de la forme de la classe (inconnue) dont l'échantillon est représentatif et du nombre d'exemples.

Pour illustrer cet aspect, nous allons chercher à classifier, avec les deux noyaux précédents, une classe donnée par

$$y \triangleq \text{sign}(50 * x_1^2 + 0.5 * x_2^2 - 1)$$

avec $x = (x_1, x_2)$ et pour des données X_i de loi uniforme dans $[-1,1]^2$.

(1) Ecrire une fonction qui construit un échantillon d'apprentissage de taille $nbApp = 150$ et un échantillon de test de taille $nbTest = 500$.
(2) Déterminer les valeurs de $dPoly$ et C permettant d'obtenir la meilleur performance avec un noyau polynomiale.
(3) Déterminer les valeurs de γ et C permettant d'obtenir la meilleur performance avec un noyau Gaussien.
(4) Comparer les résultats avec ceux de l'exercice précédent. Que constatez vous?